Optimal algorithms for sequencing of aircraft arrivals and departures

Prakash, Rakesh

2019

Prakash, R. (2019). Optimal algorithms for sequencing of aircraft arrivals and departures.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/89261>

<https://doi.org/10.32657/10220/47683>

Optimal Algorithms for Sequencing of Aircraft Arrivals and Departures

Rakesh Prakash

School of Mechanical and Aerospace Engineering

A thesis submitted to the Nanyang Technological University in
partial fulfillment of the requirement for the degree of

Doctor of Philosophy

NTU Singapore

February 2019

For Neha and Saanvi

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

15-Feb-2019

.....
Date

.....
Rakesh Prakash

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

15-Feb-2019

.....

Date

.....

Rajesh Piplani

Authorship Attribution Statement

This thesis contains material from paper(s) published in the following peer-reviewed journal(s) and conference proceeding(s) where I was the author and/or corresponding author.

Chapter 3 is published in conference proceedings as Desai, J. and Prakash, R. (2016). *Flight sequencing and scheduling: A data-driven approach*. In Proceedings of the Industrial and Systems Engineering Research Conference. Institute of Industrial Engineers, Anaheim, CA.

The contributions of the co-authors are as follows:

- A/Prof Desai provided the initial project direction.
- I formulated the algorithm. It was reviewed by A/Prof Desai.
- I and A/Prof Desai designed the computational experiments.
- I performed all the coding and conducted the computational experiments.
- A/Prof Desai reviewed the computational experiments.
- I prepared the manuscript draft and A/Prof Desai edited the manuscript drafts.

Chapter 4 is published as Prakash, R., Piplani, R., and Desai, J. (2018). *An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput*. Transportation Research Part C: Emerging Technologies, 95:570-581.

The contributions of the co-authors are as follows:

- A/Prof Desai provided the initial project direction.
- I formulated the model and the algorithm. It was reviewed by A/Prof Desai for arrival traffic.
- Assoc/Prof Piplani helped in designing the proof of the algorithm.
- Assoc/Prof Piplani helped in reviewing the algorithm for mixed-traffic scenario.
- I and/Prof Desai designed the computational experiments.
- I performed all the coding and conducted the computational experiments.
- Assoc/Prof Piplani helped in further refining the computational experiments.
- A/Prof Desai and Assoc/Prof Piplani reviewed the computational experiments.
- I prepared the manuscript draft.
- Assoc/Prof Piplani edited the manuscript draft.

Chapter 5 is published in conference proceedings as Prakash, R. and Desai, J. (2017). *A data-splitting algorithm for flight sequencing and scheduling on two runways*. In Proceedings of the Industrial and Systems Engineering Research Conference. Institute of Industrial Engineers, Pittsburgh, USA.

The contributions of the co-authors are as follows:

- A/Prof Desai provided the initial project direction.
- I formulated the model and the algorithm. It was reviewed by A/Prof Desai.
- I and A/Prof Desai designed the computational experiments.
- I performed all the coding and conducted the computational experiments.
- A/Prof Desai reviewed the computational experiments.
- I prepared the manuscript draft and A/Prof Desai edited the manuscript draft.

Chapter 6 is published in conference proceedings as Desai, J. and Prakash, R. (2016). *An optimization framework for terminal sequencing and scheduling: The single runway case*. In Proceedings of the Complex Systems Design & Management 2016 Asia Conference, pages 195-207. Springer, Cham.

The contributions of the co-authors are as follows:

- A/Prof Desai provided the initial project direction.
- I formulated the model. It was reviewed by A/Prof Desai.
- I and A/Prof Desai designed the computational experiments.
- I performed all the coding and conducted the computational experiments.
- A/Prof Desai reviewed the computational experiments.
- I prepared the manuscript draft and A/Prof Desai edited the manuscript draft.

15-Feb-2019

.....
Date

.....
Rakesh Prakash

Acknowledgements

First and foremost, I would like to thank my adviser Professor Rajesh Piplani, who guided my journey of PhD showing a lot of patience, support, and trust. Thanks for instilling confidence in me and inspiring for perfection along the way. It has been a privilege working with you.

A special thank is also due to Professor Jitamitra Desai, who introduced me to this exciting area of research. Sincere gratitude to my thesis advisory committee members Professors S. Vishwanathan and Chen Songlin for their direction during the course of my thesis. A big thank to Air Traffic Management Research Institute, for the interesting sessions, discussions, and support which helped in shaping up this thesis.

Just as importantly, I want to thank you Mummy and Papa for always standing with me and showing continuous support throughout my journey. Your values and wisdom have shaped up every aspect of my life. To my dear wife Neha, for your constant love and all the sacrifices you made for me along this journey. This couldn't have been possible without you. To Saanvi, my little angel for being an epitome of happiness for me. To my dear friend Ranjan Singh for having answers to all my questions, and solutions to all my problems. To one of my bestie Manoj for standing with me in harder times.

Last but not the least, a huge thank to all my friends and well-wishers: Rohit, Manish, Anil, Yogesh, Anish, Govind, Mayank, Prem, Shawn, Marvin, and many, many more. Thank you, I look back with great fondness upon the times we have shared.

Abstract

In airport operations, runways are scarce and expensive resources and constitute a key bottleneck. Further, capacity on the runway is limited by the safety separation between flights, thereby providing a huge opportunity to optimize the aircraft sequencing and scheduling operations on the runways, commonly referred to in the literature as the aircraft sequencing problem (ASP). However, ASP is an NP-hard problem and sequencing and scheduling decisions have to be made in real-time. Thus, motivated by the challenge posed by this very important problem, in this dissertation comprising of four essays pertaining to four different variants of ASP, models and algorithms are proposed to enable optimal sequencing in real-time. This thesis considers the ASP on a single runway, as well as, a dual-independent runway system under both segregated and mixed mode of operations. In each of the four essays, ASP is formulated as a 0–1 mixed-integer program while taking into account several realistic constraints including airline equity considerations, implemented using constrained position shifting.

The first essay addresses the ASP on a single runway with the objective of minimizing the total delay in the system. Observing the computational difficulty in getting real-time optimal solutions of various large-scale realistic instances of this problem by directly employing the MIP model, a novel *data-splitting algorithm* is proposed, the first method to solve these instances in real-time. Under the data-splitting algorithmic framework, which is also pleasingly parallel, the given set of flights is divided into several subsets, each of which is then optimized using 0–1 MIP. The proposed algorithm is based on a divide-and-conquer strategy, and novelty is introduced in both dividing and solving so as to ensure the optimality of the entire set. Detailed algorithmic insights and analysis to validate the rationale, proof of optimality, and strength of the proposed method, including a combinatorial exposition of the estimated reduction in the search space, are also presented. Computations demonstrate that the average solution

time can be reduced significantly (by as much as 99%) over direct solution by a commercial solver, while achieving the optimal solution in real-time for nearly all the instances.

The second essay considers the ASP with the objective of maximizing the total throughput of a runway. This problem enjoys a separable structure under the given constraints, thereby attracting a state-of-the-art algorithm based on dynamic programming; however, this algorithm can handle arrival traffic operations in real-time, but may not handle mixed mode of operations and some other set of problem scenarios with the same efficiency. Recognizing this, the data-splitting algorithm is tailored to solve this problem which also exploits its separability property. Further, a novel algorithmic enhancement procedure to further reduce the solution time, and a framework to split the flight dataset at multiple points to handle very large-sized instances, are presented. Computational results show that the difficult instances can be optimally solved in real-time under various scenarios, while achieving significant reduction in computational time (by as much as 99%) over direct use of the commercial solver. Further reduction in computational time is also demonstrated using the parallelized implementation.

The third essay considers the ASP on a dual-independent runway system with the objective of maximizing the total throughput of the runways. This problem is computationally harder than single runway scheduling due to the additional runway allocation decision; moreover, unlike maximizing throughput on a single runway, it does not possess any special separable structure. The lack of separability precludes the use of dynamic programming. Recognizing that large scale realistic instances of this problem cannot be solved optimally in real or near-real time by directly employing the MIP formulation either, the data-splitting algorithm is adapted to solve the problem. In addition, various algorithmic improvement procedures and a framework to split the flight dataset at multiple points, are proposed. Computational results show a significant reduction in average solution time (by as much as 92%) compared to direct use of commercial solver, while achieving optimality in all of the instances. The results indicate that sequential implementation of the algorithm solves all the instances in real or near-real time for the maximum position shifting (MPS) parameter of one, whereas the parallelized implementation solves nearly all of the instances in real or near-real time for MPS parameter value of up to two.

The fourth essay addresses the ASP over the entire terminal maneuvering area (TMA) for a single runway operating in segregated and mixed-mode scenario. In contrast with existing approaches that only consider the runway as a bottleneck, this approach optimizes flight sequences and schedules by taking into account the configuration and associated constraints of the entire TMA region. Computational results on illustrative examples show that significant reduction in delay can be achieved over the sequencing policy which only considers safety separation at runway.

Table of contents

List of figures	xix
List of tables	xxi
1 Introduction	1
1.0.1 Research Objectives	5
1.0.2 Outline of the Thesis	5
2 Literature Review	7
2.1 Problem Variants	7
2.1.1 Early Landings	7
2.1.2 Time-windows	8
2.1.3 Constrained Position Shifting	9
2.1.4 Runway Operating Mode	9
2.1.5 Problem Objectives	10
2.2 Solution Approaches: Single Runway	11
2.2.1 Dynamic programming	11
2.2.2 LP-based Branch-and-Cut	12
2.2.3 Heuristics	14
2.3 Solution Approaches: Multiple Runway	14
2.4 TMA sequencing	15
2.5 Research Gaps	16
2.6 Summary	17
3 Aircraft Sequencing on a Single Runway: Development of Data-Splitting Algorithm	19
3.1 Introduction	19

3.2	Optimization Problem	21
3.3	Data-Splitting Algorithm	24
3.3.1	DS-ASP for Arrival Traffic	25
3.3.2	DS-ASP for Arrival Traffic: Insights and Analysis . . .	31
3.3.3	DS-ASP for Arrival Traffic: Enhancements	33
3.3.4	DS-ASP algorithm: Extension to Mixed-Traffic	34
3.4	Computational Results	38
3.4.1	Effect of γ	39
3.4.2	Parallelization	41
3.5	Summary	42
4	Aircraft Sequencing on a Single Runway to Maximize Throughput:	
	A Data-Splitting Approach	45
4.1	Introduction	45
4.2	Optimization Problem	47
4.3	Implementation of DS-ASP	49
4.3.1	Solving the instance pairs	50
4.3.2	Algorithmic Enhancements by fathoming “sibling” in- stance pairs	51
4.3.3	Multiple Splits	53
4.4	Computational Results	54
4.4.1	Arrival Traffic Only	55
4.4.2	Mixed Traffic	56
4.4.3	DS-ASP in case of Unconstrained Position Shifting . . .	57
4.5	Summary	58
5	Aircraft Sequencing on Two Runways: A Data-Splitting Approach	61
5.1	Introduction	61
5.2	Optimization Problem	63
5.3	Implementation of DS-ASP	65
5.3.1	Algorithmic outline for arriving traffic	66
5.3.2	Algorithmic Enhancements and Improvements	73
5.3.3	Multiple Data-Split	78
5.3.4	Extending the Algorithm to Mixed Traffic	80
5.4	Computational Results	82

5.5	Summary	86
6	Aircraft sequencing over entire TMA	89
6.1	Introduction	89
6.2	Modeling and Analysis	91
6.3	Computational Results	95
6.4	Summary	97
7	Conclusions	99
7.1	Summary	99
7.2	Future Research Opportunities	100
7.2.1	Enhanced TMA sequencing and scheduling	100
7.2.2	Incorporating other operational constraints and objective functions	101
7.2.3	Extending the DS-ASP for multiobjective ASP	101
7.2.4	Enhancing the performance of DS-ASP	102
7.2.5	Extending the DS-ASP to other scheduling problems	102
7.2.6	Comparison of DS-ASP with metaheuristics	102
	References	103
	Appendix A Time-windows Tightening	109
	Appendix B Search Space Reduction	111
B.1	Combinatorial Exposition of Search Space Reduction	111
B.2	Reduction in B&B nodes exploration	113
	Appendix C Pseudo-Code for multiple-split	115
	Appendix D Publications Resulting from this Thesis	117

List of figures

1.1	Illustration of the relationship between delay, demand and system capacity	2
1.2	Makespan comparison between FCFS (upper) and optimal sequence (lower) of three aircraft (see Table 1.2 for separation values)	4
2.1	Delay saving of 33.3% in an example data when early landings are allowed	8
2.2	Illustrating case of $k = 1$, sequence B is not feasible as aircraft 1 and 3 are shifted by 2 units from its FCFS position	9
2.3	Schematic representation of TMA	16
3.1	Optimal schedules corresponding to three different makespan of the leading set. Solid dot represents heavy, squared boxes represent large, and hollow dots represent small aircraft	28
3.2	Rate of decrease of delay of a sequence of five aircraft in early landing scenario (δ represents the decrease in makespan, s_i and $s_i \pm \delta$ represents initial and final states of aircraft i , and different types of dots represent different aircraft type.)	32
3.3	Effect of γ on success and efficiency of algorithm in case of mixed traffic	40
3.4	Optimal delay and time for each instance pair for an instance of size of 30	42
3.5	Optimal delay and time for each instance pair for an instance of size of 40	42
4.1	Illustration of makespan decrease with increasing k	58

4.2	Illustration of solver time increase with increasing k	58
5.1	The landing order (right to left) of the last aircraft on second runway is second in the left figure whereas it is third in the right figure	68
5.2	Pareto frontier consisting of $\check{z}_1$ and $\check{z}_2$	70
5.3	Iteration 1: $(\check{z}_1, \check{z}_2) = (200, 180)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$	73
5.4	Iteration 2: $(\check{z}_1, \check{z}_2) = (220, 140)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$	73
5.5	Iteration 3: $\delta - Opt_B$ run with $(\check{z}_1, \check{z}_2) = (230, 100)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$	73
5.6	Sequence change in case of widely spaced earliest times (the time-window of each flight is indicated with oval bracket)	75
5.7	Sequence change in case of narrow time-windows, the time-window of each flight is indicated with oval bracket	76
5.8	Effect of splitting on computation time for arrival traffic of size 30 in case of $k = 1$ (IAT represents γ)	86
5.9	Effect of splitting on computation time for arrival traffic of size 30 in case of $k = 2$	86
6.1	A graphical representation of a TMA region with four entry fixes, four merge points, and a single runway	91
6.2	In figure 6.2a, when flight j reaches the node p , flight i must be at a safety separation of Δs_{ij} distance units or $\frac{\Delta s_{ij}}{v_i^e}$ time units. In figure 6.2b, when the leading flight i reaches the node q , flight j must be Δs_{ij} distance units behind or $\frac{\Delta s_{ij}}{v_j^e}$ time units.	94
6.3	Optimal sequences obtained by the three sequencing models for the case of $k = 1$	96
6.4	Optimal sequences obtained by the three sequencing models for the case of $k = 3$	96

List of tables

1.1	Table of Acronyms	2
1.2	Required time-based separations enforced at the runway threshold as per aircraft type and operating mode (Balakrishnan and Chandran 2010)	4
3.1	Computational times for large-scale realistic instances on direct implementation of ASP	24
3.2	Computational time comparison between DS-ASP and ASP where superscript a denotes arrival and m denotes mixed traffic	39
4.1	Comparison of delay with and without minimizing delay as an objective function component	46
4.2	Computational times based on a direct implementation of Problem ASP on large-scale instances, time-limit is set as 3600 seconds	49
4.3	Computational time comparison for DS-ASP, ASP, and DP (– indicates DP is better)	55
4.4	Computational time comparison of DS-ASP, its parallel implementation, and DP	55
4.5	Computational time comparison of DS-ASP with one and two splits vs DP	56
4.6	Computational time comparison of DS-ASP and ASP	56
4.7	Computational time comparison of DS-ASP and its parallel implementation	56
5.1	Computational times (sec) based on a direct implementation of Problem 2-ASP on large-scale instances	65

5.2	Effect of MPS parameter k on makespan averaged over 10 instances for each combination of instance size and γ , $k = 0$ denotes FCFS	83
5.3	Computational time comparison between DS-ASP and 2-ASP with sequential implementation, where superscript a denotes arrival and m denotes mixed	83
5.4	Computational time comparison between DS-ASP and 2-ASP with parallel implementation	84
5.5	Effect of γ on computation time for instance size 30 averaged over five instances for each γ	85
5.6	Effect of γ on computation time for instance size 40 averaged over five instances for each γ	85
6.1	Distance-based TMA separation standards (in nautical miles) enforced for arrivals	90
6.2	Computational times for large-scale realistic instances on direct implementation of ASP-TMA	97
B.1	B&B nodes comparison between DS-ASP and ASP	113

Chapter 1

Introduction

A steep increase in air traffic is forecasted over the next two decades, with air traffic worldwide doubling by 2030 if the projected annual growth rate of 4.5% persists, and global fleet size to double by 2035 (Airbus 2016). However, at major airports, the present ATM (see Table acronyms for the acronym) system itself is capacity constrained leading to vast delays. According to the Bureau of Transportation Statistics, in 2017, nearly one-fifth of all flights in the United States arrived late at their destination by at least 15 minutes, and of these late arrivals, nearly one-fourth were delayed due to the inability of the ATM system to handle the traffic, and an additional one-third were indirectly linked to these delays as a concatenated effect of an arriving aircraft's delay on its next scheduled departure. The impact of this delay is severe, directly costing billions of dollars to the air transportation industry (Airlines for America 2017) in form of fuel cost, airport charges, crew charges, passenger inconvenience cost etc. Further, the cost will only increase in the future if system capacity is not expanded. As illustrated in Fig 1.1 (Ball et al. 2010), for the future demand D_3 , delay with the current system capacity can be much higher due to a non-linear increase in delay with demand; however, increase in system capacity would lower this rate of change in delay.

Fig. 1.1 Illustration of the relationship between delay, demand and system capacity

Table 1.1 Table of Acronyms

Acronym	Meaning
ASP	Aircraft Sequencing Problem
ATSP	Asymmetric Traveling Salesman Problem
ATM	Air Traffic Management
CPS	Constrained Position Shifting
DP	Dynamic Programming
DS-ASP	Data-Splitting Algorithm for ASP
ELW	Earliest Landing time Windows
ETA	Estimated Time of Arrival
ETD	Estimated Time of Departure
FAA	Federal Aviation Administration
FCFS	First-Come-First-Served
MPS	Maximum Position Shifting
MIP	Mixed Integer Program
NM	Nautical Miles
RHC	Receding Horizon Control
TMA	Terminal Manoeuvring Area

The capacity within a radius of about 50 NM from an airport, also known as TMA is the most constrained resource within an ATM system (Arkind 2004, Bianco et al. 1997); in the event of high traffic volume, the lack of capacity in the TMA may lead to congestion, thereby causing delays. Terminal-area congestion accounted for 13% of total delayed operations in 2005 at the 35 busiest airports in USA; that has risen to 21% in 2017 (Federal Aviation Administration 2018). While injecting additional capacity into the ATM system through infrastructure development can relieve the stress placed on the system, over the short term the existing resources can be more efficiently operated by managing the bottleneck

operations related to arrivals, departures, runways, and taxiways (Anagnostakis et al. 2001, Soomer and Franx 2008). Commonly referred to in the literature as the ASP, it entails optimally sequencing and scheduling flight operations on runways which constitute a key bottleneck in the TMA of an airport (Idris et al. 1998).

Specifically, in the *static version* of ASP, given a set of aircraft, along with information on the earliest/latest operation time for each aircraft (be it an arrival or a departure), and the minimum safety separation to protect trailing aircraft from wake vortices generated by the leading aircraft; the aim is to determine a sequence and corresponding schedule of aircraft in order to optimize a desired (predefined) objective or outcome. Usually, air traffic controllers follow the FCFS rule for sequencing flight arrivals and departures based on their ETA or ETD on the *runway*. But, it is well-known that such a sequencing rule is very inefficient in practice as it induces a lot of delay into the system (Bennell et al. 2011), thus necessitating an alternate sequencing procedure which allows for the position shifts with respect to the FCFS sequence. Fig 1.2 illustrate the benefit of alternate sequencing over FCFS where 159 seconds of capacity is saved in case of former. However, the constantly changing aircraft states cause the terminal area environment to be dynamic and uncertain, and any alternate sequencing and scheduling process needs to be executed quickly whenever new events change the state of the system. Many air traffic management advisory systems such as COMPAS (Volckers 1990), MAESTRO (Garcia 1990), and FAST (Davis et al. 1994) have been developed to assist controllers in sequencing aircraft by allowing for position shifting of aircraft with respect to the FCFS sequence, but they rely on heuristics to do so.

It is clear that the development of computationally efficient models and algorithms for optimally solving the ASP in practical scenario is necessary, the challenge really lies in simultaneously achieving *safety*, *efficiency*, and *equity* in the ATM system, where: (i) passenger safety is achieved by maintaining the required separation between aircraft (Table 1.2); (ii) system efficiency is equivalent to achieving low average delay or high runway throughput; and (iii) airline equity is implemented by a CPS strategy wherein an aircraft cannot be shifted by more than k positions (the so-called MPS parameter) from its initial FCFS-based position. The equivalence of this ASP to the NP-Hard ATSP (Psaraftis 1980) and job shop scheduling problem with sequence dependent set-up times (Beasley

Fig. 1.2 Makespan comparison between FCFS (upper) and optimal sequence (lower) of three aircraft (see Table 1.2 for separation values)

et al. 2000, Bianco et al. 1997) has already been established, making real-time optimal solutions to the ASP a difficult task.

Table 1.2 Required time-based separations enforced at the runway threshold as per aircraft type and operating mode (Balakrishnan and Chandran 2010)

Aircraft Type \ Operating Mode	Departure → Departure			Departure → Arrival		
	Heavy	Large	Small	Heavy	Large	Small
Leading/Following						
Heavy	90	120	120	60	60	60
Large	60	60	60	60	60	60
Small	60	60	60	60	60	60
Aircraft Type \ Operating Mode	Arrival → Departure			Arrival → Arrival		
	Heavy	Large	Small	Heavy	Large	Small
Leading/Following						
Heavy	75	75	75	96	157	196
Large	75	75	75	60	69	131
Small	75	75	75	60	69	82

It is worth mentioning here that a fast algorithm to solve the ASP in a static setting, i.e., without taking parameters uncertainties into account, can also be applied in the dynamic and uncertain environment under the framework of RHC (Hu and Chen 2005b). Under this framework, to solve the ASP in a dynamic environment, the time axis is divided into various operating intervals, and an optimization model is applied on the predicted traffic over each of these operating intervals. Only the optimal schedule corresponding to the first interval is fixed; the time-horizon rolls over onto the second interval, and the process is repeated over subsequent intervals until the final interval solution is obtained.

1.0.1 Research Objectives

The main goal of this dissertation is to investigate ASP which is a very important problem as advances in aircraft sequencing can significantly impact the airport capacity. Even though this problem is not new and many contributions in describing, modeling, and solving it, have been recorded in the literature, this thesis aims to solve it in real-time designing novel solutions while taking advantage of the capability of commercial solvers.

Our investigation includes identifying the important variants of ASP, their modeling, characterization (equivalently, identification of special structures in the problem), and development of *tractable* and *robust* solution procedures over existing approaches. It also encompasses the demonstration of the effectiveness of the proposed solution procedures in handling realistic instances and their comparison with existing state-of-the-art methods or direct use of a commercial solver. Note that here tractability refers to the ability to optimally solve large-scale realistic instances in times that are appropriate for the application under consideration—not usual polynomial solvability; whereas robustness refers to tractability over different problem scenarios. Further note that, large scale realistic instances (in this research) refers to peak-traffic instances spanning up to about one hour. Optimality of any size of problem in real-time cannot be achieved as this is an NP-hard problem; moreover, for the purposes of this research it is reasonable to solve for this size of traffic only as for longer duration uncertainty in the arrival times of aircraft increases.

Additionally, since ASP is equivalent to ATSP, this research can be used to address the issue of computational complexity and algorithmic efficiency of various other scheduling problems, thereby contributing to the field of combinatorial optimization.

1.0.2 Outline of the Thesis

Chapter 2 presents important contributions to the ASP literature, and outlines the research gap. More specifically, visiting through all the variants of ASP, this chapter defines its most realistic version and the existing gap in solving it to optimality.

Chapter 3 introduces a novel *Data-Splitting Algorithm*, called DS-ASP, to provide the first tractable solution in the literature for the ASP on a single runway

with the objective of minimizing delay. Detailed algorithmic insights and analysis to validate the rationale, proof of optimality, and strength of the proposed method, including a combinatorial exposition of the estimated reduction in the search space, are presented. The chapter ends with the computations illustrating the efficacy of this method.

Chapter 4 extends the DS-ASP to solve the ASP on a single runway with the objective of maximizing throughput. The robustness of using this method against a state-of-the-art dynamic programming (DP) based approach is discussed. Finally, computational comparisons are made with DP and a commercial solver to demonstrate the efficacy of DS-ASP in handling alternative scenarios.

Chapter 5 investigate and formulates the ASP on a dual independent runway system, and subsequently tailors the DS-ASP to solve it. Detailed algorithmic insights and various procedures to improve the computational times are presented. At the end, computational results illustrating the efficacy of this approach are presented.

Chapter 6 extends the ASP formulation to include constraints along the entire TMA region recognizing that a bottleneck may appear anywhere within the TMA, and not just only on a runway. Further, it presents the computations to illustrate the benefit of extending the ASP to the entire TMA. Lastly, Chapter 7 summarizes the findings and presents direction for future research.

This thesis also contains four Appendices where, time-windows tightening rule (Appendix A), combinatorial characterization of the reduction in the search space (Appendix B), pseudo-code for multiple-split in case of dual-independent runway system (Appendix C), and publications resulting from this thesis (Appendix D), are presented.

Chapter 2

Literature Review

This chapter reviews the important contributions to ASP in the literature and outlines the research gap. More specifically, visiting through all the variants of ASP that have been addressed in the literature, this chapter outlines its most realistic version and the existing gaps in solving it to optimality. Further, since ASP on a single runway and multiple runways are computationally different problems, their solution approaches are discussed separately. Finally, the macroscopic version of ASP is studied which now considers entire TMA by imposing safety constraints along way-points and airways in addition to runways.

2.1 Problem Variants

There are many variants of the ASP depending upon the mode of runway operation, problem objectives, and constraints, such as the inclusion of time-windows, the width of time-windows (narrow or wide), permissibility of early landings, and CPS requirements. The inclusion of each (or a combination of) of these factors results in a significantly different problem.

2.1.1 Early Landings

When an aircraft is allowed to land *before* its target time qualifies as an *early landing* (or *time advancement*), and is found to be a very important factor that influences the total delay in the system (Lee and Balakrishnan 2008). Fig 2.1 illustrate its benefit where delay savings (including earliness and tardiness) of 33.3% seconds is observed with an early landing of 60 seconds. When early

landings are feasible, the anticipated decrease in delay (and the associated savings in fuel consumption) is accomplished at the expense of extra fuel that is consumed in speeding up an aircraft from its optimal velocity profile (which in turn dictates its target time of arrival); it may often not be economical to move the target time forward by more than a minute (Neuman and Erzberger 1991). In the ASP literature, early landings have been considered by Balakrishnan and Chandran (2010), Beasley et al. (2000), Briskorn and Stolletz (2014) whereas Bianco et al. (1997), Ghoniem et al. (2014), Harikiopoulo and Neogi (2011), Lieder et al. (2015), Lieder and Stolletz (2016), Sölveling and Clarke (2014) have ignored this factor in their work. Early landings are found to have significant computational implications for certain objective functions such as minimizing delay and cost in the case of dynamic programming (Lieder et al. 2015).

Fig. 2.1 Delay saving of 33.3% in an example data when early landings are allowed

2.1.2 Time-windows

An associated concern of early landings and a very important variant from a computational perspective is time-windows, which specify the permissible time interval within which an aircraft is allowed to arrive or depart (Ascheuer et al. 2001). Let E_i, T_i, L_i be the respective earliest, target and latest time of any aircraft i . Clearly, for any aircraft i , if early landings are allowed, its time-window is given by $[E_i, L_i]$; it is shortened to $[T_i, L_i]$ in the absence of early landings. Note that for a given set of flights, by exploiting the structure of time-windows, precedence relationships between leading and the following aircraft can be enforced; these strategies are found to be highly effective in speeding up the computational convergence process to determine an optimal solution to ASP. Usually, time-windows are *wide* in practice as an aircraft is capable of landing up to 30 minutes beyond its target time due to mandatory (excess) fuel carried on board (to account

for emergency situations). Time-windows are also found to considerably affect the computational complexity in case of dynamic programming (Balakrishnan and Chandran 2010).

2.1.3 Constrained Position Shifting

(Dear 1976) observes that in a highly dynamic air traffic environment, it is impractical to allow arbitrary deviations from the FCFS sequence for two reasons: (i) the flexibility in reordering aircraft is limited due to practical ATM system considerations (such as a controller's workload); and (ii) large deviations are unfair and unacceptable from an airline equity perspective. These observations lead to the constrained position shifting (CPS) approach, which restricts a particular aircraft in the final schedule to be within ' k ' positions of its nominal FCFS order (see Fig 2.2 for illustration). Even though k imposes limited flexibility, in general, the aircraft sequencing problem has exponential complexity (Carr 2004) as the number of possible sequences grows exponentially with k (Krafft and Schaefer 2002). Balakrishnan and Chandran (2010), Furini et al. (2014), Psaraftis (1980) have incorporated such CPS requirements in their work— only Psaraftis (1980)'s work includes it in case of multiple runways. Furthermore, modern-day real-time air traffic schedulers such as MAESTRO (Garcia 1990) also execute scheduling operations taking CPS into account.

Fig. 2.2 Illustrating case of $k = 1$, sequence B is not feasible as aircraft 1 and 3 are shifted by 2 units from its FCFS position

2.1.4 Runway Operating Mode

The operating mode of a runway can be either *segregated* or *mixed*. Segregated mode refers to a runway being used exclusively for either arrivals or departures,

whereas a mixed mode refers to the runway being utilized simultaneously for both arrivals and departures. When considering only arrival traffic, it is sufficient to separate a following aircraft from its *immediately preceding* aircraft because the triangular inequality with respect to safety separation standards (Table 1.1) is always satisfied, i.e., $\Delta_{ij} + \Delta_{jk} \geq \Delta_{ik}$, for any 3-tuple of flights (i, j, k) , where Δ_{ij} denotes the minimum time separation required between the leading aircraft i and the following aircraft j . It is this property of the time-separation matrix that motivated earlier attempts to solve the ASP using the metric traveling salesman problem framework.

However, for mixed traffic consisting of both arrivals and departures, ensuring a safe separation between the following aircraft and its immediate predecessor may not be sufficient as the triangular inequality may be violated for some of the 3-tuples. E.g., let i is a heavy arrival, j is a large departure, and k is a small arrival, then $\Delta_{ij} + \Delta_{jk} = 75 + 60 = 135$ whereas $\Delta_{ik} = 196$; clearly, triangular inequality is violated. Hence, a solution approach predicated entirely on immediate separations proves to be inadequate under a mixed-mode of operation as seen in Bianco et al. (1997), Briskorn and Stolletz (2014), Lieder et al. (2015), Psaraftis (1980). However, Balakrishnan and Chandran (2010), Beasley et al. (2000), Ghoniem et al. (2014) have successfully modeled the ASP, even under the mixed traffic scenario, by adopting more generic variable definitions and formulations rather than an immediate precedence approach.

2.1.5 Problem Objectives

Primarily, four different types of objective functions have been proposed in the ASP literature, namely: (i) Maximize throughput, or equivalently, minimize the completion time of operation of the last aircraft in the sequence (Balakrishnan and Chandran 2010, Ghoniem et al. 2014, Harikiopoulo and Neogi 2011, Psaraftis 1980); (ii) Minimize total delay (Balakrishnan and Chandran 2010, Bianco et al. 1999; 1997, Hu and Chen 2005b); (iii) Minimize total weighted delay (Beasley et al. 2000, Psaraftis 1980); and (iv) Minimize total cost (Avella et al. 2017, Balakrishnan and Chandran 2010, Briskorn and Stolletz 2014, Ernst et al. 1999, Furini et al. 2014), where cost is the function of aircraft type and its landing time. More importantly, weighted or cost versions are typically easier to solve as the

objective functions can be scaled appropriately to eliminate the more complex scheduling scenarios.

2.2 Solution Approaches: Single Runway

Both heuristics and exact approaches have been adopted in equal measure for solving the ASP. Furthermore, since it is a variant of NP-hard problem, the exact solution methodology is either based on *dynamic programming (DP)* or *branch-and-bound*. First, exact methods are elaborated and then heuristics are visited.

2.2.1 Dynamic programming

The separability requirement for DP makes it restrictive for some objective functions as continuous variable time has to be included in state definition for certain objective functions to ensure separability, causing the exponential growth in states (Psaraftis 1978). The application of this approach in the literature and its complexity w.r.t different objectives is analyzed next:

- 1. Maximizing Throughput:** ASP with the objective of maximizing throughput results in a number of states with polynomial-time complexity in n and exponential in MPS parameter k , and this complexity remains unaffected by time-windows (Balakrishnan and Chandran 2010, Balas and Simonetti 2001).
- 2. Minimizing Delay:** In presence of CPS constraints, separability can be achieved with polynomial-time states (for a particular k) in the absence of time-windows; however, this becomes computationally prohibitive in presence of time-windows (Balakrishnan and Chandran 2010).
- 3. Minimizing Cost:** In this case, (Balakrishnan and Chandran 2010) obtain polynomial bound after discretizing the time windows in a certain interval in presence of CPS constraint. Polynomial bound has also been obtained by Briskorn and Stolletz (2014), Lieder et al. (2015) when CPS is not considered; however, their algorithm is built on an assumption that the aircraft from a particular weight class are identical.

2.2.2 LP-based Branch-and-Cut

LP-based branch-and-cut algorithms have been the most deployed solution methodology for solving ASP as: (i) they accord greater flexibility in accounting for most of the operational constraints; (ii) they don't require any special structure of the problem; and (iii) branch-and-cut employed in commercial solvers is very efficient, in comparison to DP, especially when DP suffers from an exponential increase in number of states. To formulate the ASP as an optimization problem, Abela et al. (1993) and Beasley et al. (2000) define *general precedence* variables (for ensuring safety separations in mixed traffic mode), whereas Ghoniem et al. (2014) use *immediate precedence* variables for separating only arrivals; an additional set of general precedence variables are used in case of a mixed-traffic scenario.

Beasley et al. (2000)'s Model: As the work in this thesis is built upon the model in the seminal work by Beasley et al. (2000), their model is presented next. Beasley et al. (2000) model the aircraft sequencing problem as a *mixed integer program (MIP)* considering precedence restrictions, controller workload requirements, runway workload balancing, and number of landings in a given time period. They present a formulation for both single and multiple runways under segregated and mixed modes of operations, using general precedence binary variables, as presented next.

Index Sets and Parameters

- $\mathcal{F}$: Set of all arriving and departing flights
- E_i : Earliest time of arrival (departure) of aircraft i
- T_i : Target time of arrival (departure) of aircraft i
- L_i : Latest time of arrival (departure) of any aircraft i
- Δt_{ij} : Required safety separation (in seconds) at runway threshold, if flight i is ahead of flight j (see Table 1.1)
- α_i : Deviation of an early landing flight time $(T_i - t_i)^+$
- β_i : Deviation of a late landing flight time $(t_i - T_i)^+$

- g_i : The non-negative penalty cost per unit time for flight i for landing before T_i , $g_i = 1$ if objective is minimizing delay.
- h_i : The non-negative penalty cost per unit time for flight i for landing after T_i , $h_i = 1$ if objective is minimizing delay.

Decision Variables

- $x_{ij} = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}$
- t_i = Scheduled time of arrival (departure) of flight i

$$\text{ASP-RWY 1:} \quad \text{Minimize} \quad \sum_{i \in \mathcal{F}} (g_i \alpha_i + h_i \beta_i) \quad (2.1)$$

$$\text{subject to} \quad \beta_i - \alpha_i = t_i - T_i, \quad \forall i \in \mathcal{F} \quad (2.2)$$

$$x_{ij} + x_{ji} = 1, \quad \forall i < j, (i, j) \in \mathcal{F} \quad (2.3)$$

$$t_j \geq t_i + \Delta t_{ij} - M(1 - x_{ij}), \quad \forall i, j \in \mathcal{F} \quad (2.4)$$

$$E_i \leq t_i \leq L_i, \quad \forall i \in \mathcal{F} \quad (2.5)$$

$$x_{ij} \in \{0, 1\}, \quad t_i, \alpha_i, \beta_i \geq 0 \quad \forall i, j \in \mathcal{F} \quad (2.6)$$

In the above formulation, the objective function (2.1) seeks to minimize the total delay cost; constraint (2.2) defines the earliness or lateness w.r.t target time of arrival; constraint (2.3) enforces the order precedence between flights i and j ; constraint (2.4) ensures the time-based separation requirement between flights at the runway, where $M \equiv (L_i + \Delta t_{ij} - E_j)$; constraint (2.5) maintains the scheduled time of arrival (departure) to be between the earliest and latest times for each aircraft; and finally constraint (2.6) imposes the binary restrictions on the x variables, and non-negativity restrictions on the t, α, β variables.

2.2.3 Heuristics

Due to the NP-hard nature of the ASP, a number of heuristics have also been proposed to obtain quality solutions within a reasonable time frame. Notably, Beasley et al. (2001) develop a genetic algorithm for scheduling aircraft arrivals at Heathrow airport; Pinol and Beasley (2006) develop a scatter search and bionomic population-based heuristics; Atkin et al. (2007) design a tabu search method for sequencing aircraft departures; and Hansen (2004) proposes genetic algorithms to efficiently schedule aircraft landings. Furthermore, some of the works extend the static version approach to the dynamic environment using the receding horizon framework (Furini et al. 2012, Hu and Chen 2005a, Zhan et al. 2010). Any further investigation of heuristics approach is not done as that is not the focus of this research.

2.3 Solution Approaches: Multiple Runway

While the ASP on a single runway has been extensively studied from the lens of exact solutions, the multiple runway ASP (*m*-ASP) has received scant attention; moreover, the existing works have not imposed any CPS requirements (Briskorn and Stolletz 2014, Ghoniem et al. 2015, Lieder et al. 2015) except that by (Psaraftis 1978; 1980). (Psaraftis 1978) who first studies this problem notes that *m*-ASP is a theoretically harder problem to solve than single runway ASP as additional runway allocation decisions need to be incorporated into the model formulation. Moreover, he makes two important observations. First, direct application of dynamic programming, in this case, would most likely to result in an explosion of state space as it should also include the time intervals until the next landings in the state variables. Second, extra complexity is induced in presence of CPS constraints as all the sub-sequences corresponding to each runway have to be dependent. Even though Briskorn and Stolletz (2014), Lieder et al. (2015), Lieder and Stolletz (2016) propose dynamic programming based solutions for the objective of minimizing cost under arrival only mode of operations, their method is built on the assumption that aircraft of a particular category are identical. Furthermore, Briskorn and Stolletz (2014) postulate that their solution may not outperform the Beasley et al. (2000)'s MIP due to large state

space. Noticing this, Lieder et al. (2015) drop the very important early landings considerations in their solution so as to gain the computational efficiency.

Recognizing the equivalence of this problem with VRPTW where runways are viewed as vehicles and aircraft as customers, *branch-and-price* approaches which has been used quite frequently in VRPTW (Baldacci et al. 2004, Desrochers et al. 1992, Kallehauge et al. 2005) are also studied. However, unlike Wen et al. (2005) and Ghoniem et al. (2015) where they apply this approach for scheduling on independent runways, it cannot be directly applied in presence of CPS constraints as each sub-problem belonging to a runway has to be solved independently, which is not viable under the CPS framework. Nonetheless, ASP (with CPS) for multiple runways turns out to be a more practically relevant problem than its single runway counterpart as most of the busy international airports have at least two runways. When runways are used in independent segregated mode, e.g., as in Changi airport (Singapore), single runway scheduling approach is sufficient, whereas it needs to be modified for the multiple runways scenario.

2.4 TMA sequencing

The major focus in the literature has been on solving the aircraft sequencing problem by considering safety constraints at the runway, which is often a bottleneck in the system. However, it is also recognized that bottlenecks may appear anywhere throughout the TMA region illustrated in Fig 2.3 (Bianco et al. 2006, Erzberger 1995), and thus *terminal area scheduling* has also received a limited attention in the literature (Bianco et al. 2006, D'Ariano et al. 2015, Samà et al. 2014, Wong 2000). However, there exists no holistic approach to addressing safety, efficiency, and equity. Bianco et al. (2006) do not consider the use of time-windows for scheduling arrivals or departures, which greatly limits the practical applicability of their model; Samà et al. (2014) and D'Ariano et al. (2015) do not consider CPS constraint as well as earliest time considerations in their formulations; and, Wong (2000) determines the final sequence on the runway threshold with a sequencing process which starts from sequencing the aircraft in the outermost flight segments, and for each intermediate flight segment determines a sequence respecting those previously computed for upstream segments, but is inefficient as compared to finding a sequence at the runway threshold by taking into account safety separation in all TMA segments.

Fig. 2.3 Schematic representation of TMA

2.5 Research Gaps

As seen from the study of the literature, the most important variant of ASP consists of both segregated and mixed-mode of operations and incorporates all practical constraints including CPS, wide time-windows and early landings/departures. Next, of all the objectives mentioned above, the focus of this research are on minimizing total delay and maximizing throughput as they are important from the perspective of air traffic control and directly address system capacity and efficiency (Lee and Balakrishnan 2008); the other objectives such as minimizing cost and minimizing weighted delay (with weights being number of passengers) are not considered here as they are more important from the airline perspective. Finally, with regard to this particular variant of ASP, the following research gaps emerge:

1. LP-based branch-and-cut approach (including Beasley et al. (2000)) is intractable for $k \geq 2$; the importance of this approach is the flexibility it offers in accounting for various operational constraints. Investigation shows that 0–1 MIP formulations cede to *big-M* constraints, and thereby converge slowly.
2. Existing exact methodologies are unable to provide tractable solutions in

case of one of the most important objective function of minimizing delay—even though DP (Balakrishnan and Chandran 2010) is efficient for maximizing throughput, it is unable to account for the important time-window constraints in case of minimizing delay. Further, DP can become intractable in case of former objective too when the time is included in the state function, as in case of mixed-mode traffic conditions.

3. No attempts to formulate and tractably solve the dual-independent runway sequencing and scheduling are found for the variant of ASP under consideration. This problem is originally described in (Psaraftis 1980) and has great practical implications for major airports such as Changi (Singapore), Heathrow (London) which have this runway configuration.

4. Optimization framework of ASP considering safety separation along entire TMA has not been developed for the variant under consideration.

2.6 Summary

In this chapter, the important contributions to ASP in the literature are reviewed, and research gaps identified. More specifically, this chapter underscores an emphasis towards developing tractable solution for single runway and dual runways scheduling, which should be flexible enough to account for important objective functions and operational constraints, preferably the mathematical programming based approaches. Additionally, it also emphasizes the design of an optimization framework of ASP, considering safety separations along entire TMA.

In the next chapter, ASP on a single runway with the objective of minimizing delay is investigated. Recognizing the absence of tractable approaches, a novel *Data-Splitting Algorithm*, called DS-ASP, is introduced to solve this problem. Detailed algorithmic insights and analysis to validate the rationale, proof of optimality, and strength of the proposed method, including a combinatorial exposition of the estimated reduction in the search space, are also presented.

Chapter 3

Aircraft Sequencing on a Single Runway: Development of Data-Splitting Algorithm

3.1 Introduction

Continuing on extensive literature survey for the ASP on a single runway presented in Chapter 2, this chapter addresses the variant of the problem comprising of a mixed-mode of operations, while incorporating all practical constraints including CPS, wide time-windows, early landings/departures, and the objective of minimizing total delay (earliness and tardiness). There are two main motivations behind studying this particular problem variant:

1. As per a study by Lee and Balakrishnan (2008), from the air-traffic system perspective, achieving lower delay and higher throughput is the most important goals, and those can best be achieved with the objective of minimizing delay.
2. Existing exact methodologies are unable to provide *tractable* solutions (real-time optimal solutions of large-scale realistic instances) for the variant of ASP under consideration—even though DP (Balakrishnan and Chandran 2010) is efficient for various other objective functions of ASP with CPS, it is unable to account for the important time-window considerations in case of minimizing delay.

Next, the formulation of the ASP is revisited as a 0–1 *mixed-integer program* (MIP) which is an adaptation of Beasley et al. (2000)’s model enhanced with

the inclusion of the CPS constraints and a modified objective of minimizing the total delay in the system. Recognizing that it is not computationally viable to solve this model for large-scale realistic instances, this chapter proposes a novel *data-splitting algorithm*, which is referred to as **DS-ASP**, that optimizes flight sequences by a repeated application of the 0–1 MIP on smaller data sets, and demonstrate the efficacy of the approach. Another important and intrinsic feature of the proposed data-splitting framework is that it is pleasingly parallelizable.

From modeling, algorithmic, and computational perspectives, several important contributions to the ASP literature stem from this work. The 0–1 MIP formulation presented here encapsulates the most realistic version of the ASP by simultaneously accounting for system safety, runway efficiency, and airline equity, as model constraints or via the objective function. This is the first attempt to exactly solve the large-scale realistic instances of the problem under consideration. Furthermore, it is the first LP-based approach in literature to solve ASP in presence of CPS constraints. Finally, the biggest hindrance thus far to adopting exact optimization approaches for runway scheduling in practice has been the lack of responsiveness of these algorithms to large-scale realistic instances. The remarkable speed-up exhibited by the proposed data-splitting algorithm now offers a promising avenue for such methodologies to provide real-time optimal solutions and be integrated within arrival-departure management systems.

The remainder of this chapter is organized as follows. In Section 3.2, the 0–1 MIP formulation is presented for the ASP along with some preliminary computations that serve to highlight the inherent difficulty of solving the ASP via a straightforward MIP approach. Next, in Section 3.3, the details of the data-splitting algorithm and its pseudo-code are described. Detailed algorithmic concepts and analysis are provided that offer critical insights into the development of this method, and validate the computational performance of the algorithm. Then, Section 3.4 presents detailed computational results and analysis pertaining to large-scale (realistic) instances, and finally, Section 3.5 summarizes the contributions of this chapter.

3.2 Optimization Problem

The ASP can now be formulated as a 0–1 mixed integer program, where the index sets and parameters are as defined below.

Description of Index Sets and Parameters

- $\mathcal{F}$: Set of all arriving and departing flights.
- $\mathcal{C}$: Set of all aircraft categories (or types/classes), given by $\mathcal{C} \equiv \{H: \text{Heavy}, L: \text{Large}, S: \text{Small}\}$.
- E_i : Earliest time of arrival (departure) of aircraft i .
- L_i : Latest time of arrival (departure) of aircraft i , where $L_i > E_i$.
- T_i : Target time of arrival (departure) of aircraft i , where $E_i \leq T_i \leq L_i$.
- Δ_{ij} : Mandatory minimum safety separation (in seconds) enforced at the runway threshold, if flight i is ahead of flight j (see Table 1.1).
- $seq_S(i)$: Position of flight i in any feasible sequence S .
- k : Maximum position shifting (MPS) parameter.

Decision Variables:

$$x_{ij} = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}$$

t_i = Scheduled time of arrival/departure of flight i

$$\text{ASP :} \quad \text{Minimize} \quad \sum_{i \in \mathcal{F}} |t_i - T_i| \quad (3.1)$$

$$\text{subject to: } x_{ij} + x_{ji} = 1, \quad \forall i < j, (i, j) \in \mathcal{F} \quad (3.2)$$

$$t_j \geq t_i + \Delta_{ij} - M(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{F} \quad (3.3)$$

$$i - k \leq (F - \sum_{j \in \mathcal{F}, i \neq j} x_{ij}) \leq i + k \quad \forall i \in \mathcal{F} \quad (3.4)$$

$$E_i \leq t_i \leq L_i, \quad \forall i \in \mathcal{F} \quad (3.5)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in \mathcal{F}; \quad t_i \geq 0, \quad \forall i \in \mathcal{F}. \quad (3.6)$$

In the above formulation, the objective function (3.1) seeks to minimize the total delay (earliness and tardiness); constraint (3.2) enforces the order precedence relationship between flights i and j ; constraint (3.3) ensures the time-based separation requirements between flights at the runway (as mandated by the FAA for various flight classes (see Table 1.1), where $M \equiv (L_i + \Delta_{ij} - E_j)$); constraint (3.4) imposes the CPS constraint that an aircraft cannot be shifted by more than ‘ k ’ positions from its initial (FCFS) position, where for the sake of notational convenience, $seq_{FCFS}(i) \equiv i, \forall i \in \mathcal{F}$, i.e., aircraft are labeled based on their FCFS priority order, and $F \equiv |\mathcal{F}|$; constraint (3.5) maintains the scheduled time of arrival (departure) at the runway to be between the earliest and latest times for each aircraft; and finally constraint (3.6) imposes binary and non-negativity restrictions on x - and t - variables, respectively.

The ASP formulation described above can be further improved by using variable-fixing strategies based on the CPS constraint and the well-known *earliest landing time-window* (ELW) (Briskorn and Stolletz 2014) rule. These fixing help in eliminating the x_{ij} variables in advance, thereby causing a significant reduction of the search space. As discussed below, Proposition 3.1 fixes x_{ij} variables to 1 when difference in FCFS position of two flights i, j is greater than or equal to twice the MPS parameter value. Similarly, Proposition 3.2 does the same when the associated times (E, T, L) of these two flights follow the lexicographic ordering.

Variable Fixing Strategies

Proposition 3.1. *For a pair of aircraft $(i, j) \in \mathcal{F}$, if the FCFS sequence positions of aircrafts i and j satisfy the condition: $j - i \geq 2k$, where ‘ k ’ is the MPS parameter, then $x_{ij} = 1$ can be fixed.*

PROOF: Without loss of generality, suppose that $j > i$, i.e., aircraft i is ahead of aircraft j in the FCFS sequence. Given ‘ k ’, in any sequence, aircraft i can

only be placed between $(i - k)$ and $(i + k)$ positions, and similarly the position of aircraft j can range from $(j - k)$ to $(j + k)$. If $j - i \geq 2k$, then $j - k \geq i + k$, which implies that, even in the extreme case, flights i and j maintain their nominal (FCFS) precedence order at optimality. $\square$

Proposition 3.2. (ELW Rule) *Consider a pair of aircraft $(i, j) \in \mathcal{F}$ that belong to the same category. Suppose that the time-window restrictions satisfy the following conditions:*

(i) $E_i \leq E_j$; (ii) $T_i \leq T_j$; (iii) $L_i \leq L_j$; then $x_{ij} = 1$ can be fixed.

PROOF: This is proved by contradiction. Given a pair of flights $(r, s) \in \mathcal{F}$ that satisfy the above conditions, let S^* denote an optimal sequence, and suppose that flight s is ahead of flight r at optimality, i.e., $x_{sr}^* = 1$. Let t_r^* , and t_s^* be the scheduled time of flights r and s in their optimal sequence. Let $D(S^*)$ denote the optimum (minimum) delay resulting from this optimal sequence S^* . Therefore, from (3.1),

$$D(S^*) = \sum_{i \in \mathcal{F}, i \neq r, s} |t_i^* - T_i| + |t_r^* - T_r| + |t_s^* - T_s|, \text{ where } t_s^* < t_r^*.$$

Moreover, by virtue of optimality, $D(S^*) \leq \sum_{i \in \mathcal{F}} |t_i - T_i|$, for any feasible sequence of flights.

Now, construct a new sequence, say S_{new} , where flights r and s swap their optimal positions, while all the remaining flights retain their optimal S^* sequence positions. Now, as r and s belong to the same category, there is no change in the scheduled times for all other flights, except r and s . Evaluating the total delay for this new sequence S_{new} ,

$$D(S_{new}) = \sum_{i \in \mathcal{F}, i \neq r, s} |t_i^* - T_i| + |t_r^{new} - T_r| + |t_s^{new} - T_s|.$$

But $t_r^{new} = t_s^*$ and $t_s^{new} = t_r^*$, which yields,

$$\begin{aligned} D(S_{new}) &= \sum_{i \in \mathcal{F}, i \neq r, s} |t_i^* - T_i| + |t_s^* - T_r| + |t_r^* - T_s| \\ &< \sum_{i \in \mathcal{F}, i \neq r, s} |t_i^* - T_i| + |t_s^* - T_s| + |t_r^* - T_r| \quad (\text{when } T_r \leq t_s^* < t_r^* \leq T_s) \\ &= D(S^*), \end{aligned}$$

which is a contradiction. Hence, $x_{rs} = 1$ preserves the optimality, proving the assertion. $\square$

To gain an understanding of the computational difficulties involved in solving ASP, the optimization model given by (3.1)–(3.6), enhanced with the variable fixing strategies outlined in Propositions 3.1-3.2 is tested on several randomly

generated large-scale (realistic) instances. From the results recorded in Table 3.1, it can be clearly seen that the computational time increases exponentially¹ with number of flights and the MIP model fails to converge to optimality in several instances, within a preset time-limit of 3600 seconds.

Table 3.1 Computational times for large-scale realistic instances on direct implementation of ASP

Instance	type	# of flights	MPS (k)	
			2	3
1	Arrival	30	113	1425
2	Mixed	30	262	3171
3	Arrival	40	2610	> 3600
4	Arrival	40	1141	> 3600
5	Mixed	40	> 3600	> 3600
6	Mixed	40	> 3600	> 3600

As the ASP model presented above cannot handle large-scale instances efficiently, a novel data-splitting algorithm is proposed which involves solving multiple ASP on relatively smaller data-sets using realistic instances.

3.3 Data-Splitting Algorithm

The proposed *data-splitting algorithm (DS-ASP)* essentially involves dividing the original flight data-set into several pairs of *smaller sized* sets of leading and following aircraft, determining the optimal solution for the corresponding leading and the following sets, and finally, comparing the solutions of all the pairs to select the overall optimum. The structure of the data-splitting algorithm exploits three key observations: (i) Smaller data-sets are computationally more efficient to solve than bigger data-sets; (ii) As a result of the CPS constraint, the number of pairs of leading and following data-sets are limited, to the order of $\binom{2k}{k}$, where ' k ' is the value of the MPS parameter, and *independent* of the total number of flights; and (iii) reduction in search space when solved this way, as illustrated in the combinatorial exposition (Appendix B). Throughout this chapter, the discussion is restricted to realistic instances of peak traffic with

¹In our computations, weak lower bound is observed while investigating the possible causes behind the computational complexities. The formulation is further strengthened by means of valid inequalities given in Desai and Prakash (2016), Ghoniem et al. (2014), Reformulation-Linearization-Technique (RLT) proposed in Sherali and Adams (1990; 1994), and a novel time-windows tightening rule given in Appendix A. However, the improvement in lower bound is obtained at extra computational cost, and therefore this technique is not incorporated into our model.

scheduling horizon of approximately one hour consisting of traffic size 30 to 40 in case of arrival and 30 to 50 in case of mixed traffic, an aircraft mixture consisting of a high percentage of heavy ($\geq 40\%$) and large aircraft ($\geq 40\%$), average inter-arrival time of flights to be around one minute, wide time-windows, and $k \leq 3$.

A detailed description of the algorithm is given below, where procedure for only arriving traffic is described first and subsequently extended to the case of mixed traffic.

3.3.1 DS-ASP for Arrival Traffic

Preparation of instance pairs

Let n denote the number of aircraft positioned at $\{1, \dots, m-1, m, m+1, \dots, n\}$ in FCFS order. After splitting this flight data into two (equal when n is even, else unequal) halves, the leading set A and the following set B are composed of aircraft at positions $\{1, \dots, m\}$ and $\{m+1, \dots, n\}$, respectively. Owing to the CPS constraint, as the only aircraft between $(m-k+1)$ and $(m+k)$ positions can crossover between the leading and following sets (see Proposition 3.1), there are only $\binom{2k}{k}$ such pairs of leading and following aircraft sets; this splitting or partitioning is a pure combinatorial exercise and independent to a particular sequence or flight-data. Note that both the *aircraft type* and the *arrival time* of the last scheduled flight in the leading set affect the delay of the following set because aircraft in the following set have to maintain the required safety separations from the last aircraft in the leading set. In order to account for the effect of *aircraft type*, the number of pairs is further enlarged by fixing all possible aircraft in the leading set that can be positioned last. (The effect of the *arrival time* will be accounted for in the solution stage.) Specifically, observing that only aircraft between positions $(m-k)$ to m can occupy the last position in set A , each pair (A, B) results in additional combinations, denoted as $(\hat{A}, B)$, which will finally be used by the algorithm. This instance generation scheme is illustrated with an example below.

Consider an instance with eight aircraft $\{H1, H2, H3, L4, L5, H6, S7, S8\}$, where each aircraft is represented by its aircraft type (H : Heavy, L : Large, S : Small) and its relative position in the FCFS sequence. Assuming $k = 2$, there are $\binom{4}{2} = 6$ resulting pairs (A, B) , which are given by:

1. $A = \{H1, H2, H3, L4\}, B = \{L5, H6, S7, S8\}$
2. $A = \{H1, H2, H3, L5\}, B = \{L4, H6, S7, S8\}$
3. $A = \{H1, H2, L4, L5\}, B = \{H3, H6, S7, S8\}$
4. $A = \{H1, H2, H3, H6\}, B = \{L4, L5, S7, S8\}$
5. $A = \{H1, H2, L4, H6\}, B = \{H3, L5, S7, S8\}$
6. $A = \{H1, H2, L5, H6\}, B = \{H3, L4, S7, S8\}$.

For each of the above listed pairs (A, B) , permuting flights within set A , predicated on the aircraft occupying the last position in this set (denoted as $\bullet$), results in several additional combinations $(\dot{A}, B)$. For example, when $A = \{H1, H2, H3, L4\}$ and $B = \{L5, H6, S7, S8\}$, such permutations within set A result in the following three instance pairs:

1. $\dot{A} = \{H1, H2, H3, \dot{L4}\}, B = \{L5, H6, S7, S8\},$
2. $\dot{A} = \{H1, H2, L4, \dot{H3}\}, B = \{L5, H6, S7, S8\},$
3. $\dot{A} = \{H1, H3, L4, \dot{H2}\}, B = \{L5, H6, S7, S8\}.$

Note that, for the case $A = \{H1, H2, H3, H6\}$ and $B = \{L4, L5, S7, S8\}$, as aircraft $H6$ cannot be scheduled before position 4 in any sequence (as this would violate the CPS constraint), no further pairings are feasible. Therefore, in the above example, the total number of all possible instance pairs $(\dot{A}, B)$ is twelve (three pairs of $(\dot{A}, B)$ each from first three pairs of (A, B) , and one pair each from last three pairs). Furthermore, if pair (A, B) or $(\dot{A}, B)$ violates the ELW rule (see Proposition 3.2) based on the given data, then some of the pairs, e.g. $\dot{A} = \{H1, H3, L4, \dot{H2}\}$ and $B = \{L5, H6, S7, S8\}$, can be discarded, as $H2$ lands before $H3$ using this rule, whereas, it is scheduled in a reverse order in the given pair. As a result, second, fifth, and sixth (A, B) pairs can be directly discarded ($L4$ is scheduled ahead of $L5$ in the second pair, and $H3$ is ahead of $H6$ in case of fifth and sixth pairs.) Additionally, for similar reason, pair $\dot{A} = \{H1, H3, L4, \dot{H2}\}, B = \{L5, H6, S7, S8\}$ from the first (A, B) pair and $\dot{A} = \{H1, H2, L5, \dot{L4}\}, B = \{H3, H6, S7, S8\}$ from the third, can also be discarded. Eventually, only five pairs of $(\dot{A}, B)$ instances may remain (two each from first and third and one from fourth (A, B) pair) for the solution phase of this algorithm, as compared to the originally derived twelve.

Solving the instance pairs

First, a *basic* method is briefly discussed to solve an instance pair to optima, and an *advanced* methodology is described later that will be part of the proposed algorithm. In the basic method, starting from the *completion time* of the last flight of the leading set (equivalently, the *makespan* of set $\dot{A}$) at which the leading set observes minimum delay, the delay of the leading and the following set is evaluated for each possible makespan of the leading set. (Note that since flights in the following set can land only after completion of the leading set, makespan of the leading set dictates the search space of both the leading and following sets.) Since makespan cannot be reduced below a minimum value, moreover it can only take a limited number of discrete values (not every makespan is feasible), the number of such evaluations is limited. Finally, all delays are compared to get the minimum. So, even though this method is guaranteed to be optimal and computationally efficient, a better method can be developed by analyzing the optimal data of this intrinsically complex problem.

The guiding principle of the advanced method is that when optimization is applied independently to each subset of a pair, the correct optimum can only be achieved when the individual optima are congruent to optima of the pair as these subsets are concatenated to provide the overall set. The individual optima refer to the optima of leading or the following set, whereas optima of the given pair refers to the optima of a sequence where leading aircraft and the following aircraft are same as $\dot{A}$ and B . Subsequently, if congruence cannot be confirmed from the individual optima, the optimization problems are modified so as to establish it.

For the problem under consideration, in our test cases concatenating the individual optima achieved with the objective function of minimizing delay couldn't result in the optima of a given pair. Therefore, modification of the optimization problems is attempted by first investigating the relationship between delays of the leading/following set and the makespan of the leading set. The motivation behind this is the fact that the delay of the following set can only be affected through the makespan of the leading set as the last aircraft of the latter is fixed; further, any schedule of the leading set with any objective has to end at its makespan. Now, as it dictates the upper bound (resp. lower bound) of search space of the leading set (resp. following set), a higher makespan of the leading

set may result in a lower delay for set A , it may also result in a higher delay for set B , and vice-versa.

The following example illustrates this concept. Let aircraft $\{H1, H2, L3\}$ be in the leading set and $\{H4, L5, S6\}$ be in the following set. Let the target times (in seconds) be 60, 120, 180, 240, 300, 360 respectively, where all aircraft can land early by up to 60 seconds and can be late by up to 1800 seconds. The optimization results are produced in three scenarios corresponding to three different values of makespan of the leading set. The results are also represented in Figure 3.1 and explained below.

Scenario A. *Makespan of the leading set is 261 seconds.* Delays of the leading set, the following set, and given pair are 149, 406, and 555 seconds.

Scenario B. *Makespan of the leading set is 277 seconds.* Delays of the leading set, the following set, and given pair are 133, 454, and 587 seconds. Note that search space of the leading set is larger and the following set is smaller, in comparison to scenario A, and hence delay values are lower and higher, respectively.

Scenario C. *Makespan of the leading set is 253 seconds.* Delays of the leading set, the following set, and given pair are 157, 382, and 539 seconds. Note that search space of the leading set is smaller and the following set is larger, in comparison to scenario A, and hence delay values are higher and lower, respectively.

Note that Scenario C corresponds to the optimal schedule of the given pair. Observe that delay in case of Scenario B is higher in comparison to Scenario C.

Fig. 3.1 Optimal schedules corresponding to three different makespan of the leading set. Solid dot represents heavy, squared boxes represent large, and hollow dots represent small aircraft

Next, it is found that there is a minimum rate of increase and a maximum rate of decrease of the delay of the following set with varying makespan of the

leading set. Fortunately, these minimum and maximum rates can be estimated in advance; moreover, they are equal (to $|B|$, where $|B|$ is the cardinality of set B) in case of realistic instances of arrival traffic (rationale is given in Section 3.3.2). Next, the foundation of solution design is laid. So, if these rates are equal, say to some quantity e , the congruence to overall optima can be established after including an extra term ez in the original objective function of minimizing delay of the leading set, where z is the makespan variable. Towards this end, optimum makespan of the leading set relates to minimum delay of the given pair—increasing it decreases the delay of the leading set at the rate $\leq e$, i.e., less than or equal to the minimum rate of increase of delay of the following set, and vice-versa (proof is given in 3.3.2).

Next, the rates are estimated in advance, but need to be validated from the optimal solution of the following set in order to conclude optimality of this method. Therefore, the optimization problem of set B is modified by introducing two non negative continuous indicator variables δ_1 and δ_2 , where non-zero value of δ_1 implies that minimum rate of increase of delay is $< e$ (the estimated minimum); non-zero value of δ_2 implies that maximum rate of decrease of delay is $> e$ (the estimated maximum); zero-values for both confirm optimality by validating the estimated rates. Summarizing the above discussion, the optimization problems for set A and B can be framed as:

$$Opt_A \equiv \min \left\{ \sum_{i \in A} (|t_i - T_i|) + ez; 3.2 - 3.6, z \geq t_i, \forall i \in A \right\}.$$

$$Opt_B \equiv \min \left\{ \sum_{i \in B} (|t_i - T_i|) - e\delta_1 + e\delta_2 + \varepsilon(\delta_1 + \delta_2); 3.2 - 3.6, t_i \geq t_l + \Delta_{li} + \delta_1 - \delta_2 \right\},$$

where $i \in B$, l is the last aircraft in A , and ε is a preset tolerance.

The above concept can be understood by following Figure 3.1. It can be observed that the least delay is observed in case of Scenario C (which is also the optimal delay.) Further, from Scenario A & B, it can be observed that, as the makespan of the leading set is decreased by one unit, delay of the leading set (total earliness and tardiness) increases by one unit whereas delay of the following set decreases by three units; therefore, makespan of the leading set keeps decreasing until it reaches the Scenario C, where no more reduction is possible as makespan of the leading set reaches its minimum. This phenomenon is encapsulated in the above modification of the leading set; here, the value of e is estimated to be three (cardinality of the following set; the rationale is discussed

later), whereas the delay of the leading set increases by one unit with every unit of decrease in makespan of the leading set. Hence, the minima of the leading set occur when either no further reduction is possible in the makespan of the leading set or the increase in delay becomes more than e with any further reduction; in this example, the former is observed. Next, as aforementioned, modification in the following set is aimed at validating e . Consider, for example, scenario C, where the delay of the following set is minimum. Now, with any positive change in δ_1 , the change in objective function is: $e\delta_1$ (change in delay) $-e\delta_1 + \varepsilon\delta_1$, which comes out to be $\varepsilon\delta_1$ (an increase) and hence causes δ_1 to remain zero. Similarly, $\delta_2 = 0$ in the above example. Together, it validates that $e = 3$.

A formal statement of the proposed data-splitting algorithm is listed in pseudo-code form next.

Algorithm 3.1: Pseudo-code for arrival traffic only

- 1: Set iteration counter $p \leftarrow 0$, $OptVal^{(p)} \leftarrow +\infty$, incumbent $\leftarrow OptVal^{(p)}$, $\mathcal{P} = \{\text{set of all pairs } (\dot{A}, B)\}$.
 - 2: If $\mathcal{P} = \emptyset$, STOP; return incumbent as the best solution. Else, set $p \leftarrow p + 1$, go to Step 3.
 - 3: Arbitrarily select one instance pair $(\dot{A}_p, B_p)$ from $\mathcal{P}$.
 - 4: Solve $Opt_{\dot{A}}$ for set $\dot{A}_p$, with $e = |B_p|$. Let $\dot{z}$ be the completion time.
 - 5: Solve Opt_B for set B_p , with $e = |B_p|$.
 - 6: If $\delta_1 = 0$ and $\delta_2 = 0$, concatenate the optimal sequences obtained in Steps 4 and 5, set $OptVal^{(p)} \leftarrow \sum_{i \in \dot{A}_p \cup B_p} (|t_i - T_i|)$, go to final Step 7; else, print "Algorithm fails to converge optimality".
 - 7: If $OptVal^{(p)} < \text{incumbent}$, set incumbent $\leftarrow OptVal^{(p)}$. Update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\dot{A}_p, B_p\}$. Go to Step 2.
-

REMARK: As discussed below, in important peak traffic and realistic wide-time windows conditions, it is always possible to accurately estimate e causing δ_1 and δ_2 to remain zero. However, when the highly unlikely scenario of these parameters taking non-zero value occurs: $\delta_1 > 0$ situation can be handled by switching to Algorithm 3.2 (details provided later), whereas ASP is directly solved using solver in case of $\delta_2 > 0$. Note that using solver can incur a higher computational cost.

3.3.2 DS-ASP for Arrival Traffic: Insights and Analysis

In the following discussion, the rationale is presented as to why $e = |B|$ for realistic instances of arrival traffic, and as are the proofs to validate the modifications of the optimization problems.

Rationale

Consider a scenario where all the aircraft in any feasible schedule of the following set are late (arrive after their respective target times but within their time windows), and time-windows are wider. In this scenario, it can be shown that the rate of increase of delay of the following set is $\geq |B|$ with increase in makespan of the leading set (Proposition 3.3); and (ii) rate of decrease of delay is $\leq |B|$ with decrease in makespan (Proposition 3.4). This suffices to show that $e = |B|$ for any instance pair in case of realistic instances of arrival traffic for the problem under consideration, because: (i) based on detailed analysis and substantiated by empirical observation on numerous peak traffic instances (see Section 3.4.1 for more detail), it can be inferred that all the flights in any feasible schedule of the following set arrive later than their target time; and, (ii) time-windows are wider in case of realistic scenarios.

Proposition 3.3. *Rate of increase of delay of the following set is $\geq |B|$ with increase in makespan of the leading set.*

PROOF. Clearly, when the current optimal sequence of the following set, say Π , does not change with increasing makespan, the rate of increase of delay is $|B|$. However, a new sequence, say $\bar{\Pi}$, may emerge after a considerable increase in makespan, due to possible violations in latest time of landing for some of the flights in Π . In this case, the delay calculated over $\bar{\Pi}$ must be greater than or equal to that calculated over Π , as shorter delay then can be obtained with $\bar{\Pi}$ at the current makespan too, thereby contradicting that Π is the current optimal sequence. Since higher delay increases the rate of increase, the proposition is proved. $\square$

Proposition 3.4. *The rate of decrease of delay of the following set is $\leq |B|$ with a decrease in makespan of the leading set.*

PROOF. In this case, the rate of decrease will not exceed $|B|$ for three reasons. First, when sequence of flights in set B is pulled in, early landings may be

Fig. 3.2 Rate of decrease of delay of a sequence of five aircraft in early landing scenario (δ represents the decrease in makespan, s_i and $s_i \pm \delta$ represents initial and final states of aircraft i , and different types of dots represent different aircraft type.)

observed for some of the flights, thereby causing the rate of decrease to be less than $|B|$ (illustrated in Figure 3.2, where the rate of decrease of delay is three for five aircraft as fourth aircraft lands early). Second, possible violations of earliest time of landing result in a new sequence with higher delay, thus further reducing the rate of decrease of delay. Third, since time-windows are wider (an aircraft can be late up to as much as 1800 seconds), based on analysis and empirical observations (see Section 3.4.1), it can be inferred that emergence of a new sequence with better delay will not be observed for realistic instances. Hence, the proof. $\square$

Proofs

Throughout the following discussion, let $\dot{d}_m$ (resp. d_m) denote the delay of the leading set (resp. following set) w.r.t makespan m and the estimated rate e . All other notation is standard.

Lemma 3.1. *Optima corresponding to $\dot{z}$ implies $\dot{d}_{\dot{z}} - \dot{d}_{\dot{z}+\delta} \leq e\delta$. Similarly, $\dot{d}_{\dot{z}-\delta} - \dot{d}_{\dot{z}} \geq e\delta$.*

PROOF. The optimal objective function value corresponding to $\dot{z}$ is $\dot{d}_{\dot{z}} + e\dot{z}$. Similarly, optimal objective function value corresponding to $\dot{z} + \delta$ is $\dot{d}_{\dot{z}+\delta} + e(\dot{z} + \delta)$. Since optima occurs at $\dot{z}$,

$$\begin{aligned} \dot{d}_{\dot{z}+\delta} + e(\dot{z} + \delta) &\geq \dot{d}_{\dot{z}} + e\dot{z} \\ \Rightarrow \dot{d}_{\dot{z}} - \dot{d}_{\dot{z}+\delta} &\leq e\delta \text{ (on rearranging)} \end{aligned}$$

Hence the proof. Similar proof can be given for second part of the Lemma. $\square$

Lemma 3.2. $\delta_1 > 0$ in the optimum solution of Set B implies minimum rate of increase of delay is $< e$. Similarly, it implies maximum rate of increase is $> e$ for $\delta_2 > 0$.

PROOF. The optimal objective function value corresponding to $\dot{z}$ is $d_{\dot{z}}$. Similarly, optimal objective function value corresponding to $\dot{z} + \delta_1$ is $d_{\dot{z}+\delta_1} - e\delta_1 + \varepsilon\delta_1$. Since optima occurs at $\dot{z} + \delta_1$,

$$\begin{aligned} d_{\dot{z}} &\geq d_{\dot{z}+\delta_1} - e\delta_1 + \varepsilon\delta_1 \\ \Rightarrow \frac{d_{\dot{z}+\delta_1} - d_{\dot{z}}}{\delta_1} &\leq e - \varepsilon \text{ (on rearranging)} \end{aligned}$$

The above implies minimum rate of increase in delay is less than e . Hence the proof. Similar proof can be given for second part of Lemma. (Note that ε is set very small to ensure that indicators take non-zero value for any possible small deviation in the estimated rates.) $\square$

Corollary 3.1. $\delta_1 = 0$ and $\delta_2 = 0$ in the optimum solution of Set B implies rates are as estimated.

Theorem 3.3. $\delta_1 = 0$ and $\delta_2 = 0$ is condition of optimality for any instance pair under consideration.

PROOF. From Corollary 1, $\delta_1 = 0$ and $\delta_2 = 0$ implies rates equal to e , i.e., $d_{\dot{z}+\delta} - d_{\dot{z}} \geq e\delta$, and $d_{\dot{z}} - d_{\dot{z}-\delta} \leq e\delta$, for any $\delta > 0$. From Lemma 1, $\dot{d}_{\dot{z}} - \dot{d}_{\dot{z}+\delta} \leq e\delta$, and $\dot{d}_{\dot{z}-\delta} - \dot{d}_{\dot{z}} \geq e\delta$. Combining these two results, $\dot{z}$ always observes the overall minimum delay. Hence, the proof. $\square$

3.3.3 DS-ASP for Arrival Traffic: Enhancements

The computational efficiency of the proposed DS-ASP algorithm can be enhanced further by using the information from optimal solution of a pair $(\dot{A}, B)$ to fathom the optimization of the following set of its sibling $((\dot{A}, B)$ pairs from the same (A, B) pair are considered siblings). Consider two instance pairs $(\dot{A}_1, B)$ and $(\dot{A}_2, B)$, resulting from the same parent pair (A, B) . Let $\hat{\mathcal{F}}$ represent the set of flights in set B between positions $(m+1)$ to $(m+k+1)$, and let $(\dot{z}_1, \dot{d}_{\dot{z}_1}, \dot{l}_1)$ and $(\dot{z}_2, \dot{d}_{\dot{z}_2}, \dot{l}_2)$ denote the completion time, delay, and last flight in the optimal solutions for set $\dot{A}_1$ and $\dot{A}_2$, respectively. Furthermore, suppose that Set B has been solved to optimality in pair $(\dot{A}_1, B)$, with flight $f \in \hat{\mathcal{F}}$ placed at the first position in the optimal solution of set B . Now, consider the following two cases:

Case 1. Suppose that flight f is required to be at first position in an optimal solution of set B . Calculate,

$$s = (\dot{z}_2 - \dot{z}_1) + (\Delta_{i_2f} - \Delta_{i_1f}), \text{ and } D = \dot{d}_{\dot{z}_2} - \dot{d}_{\dot{z}_1} + rs,$$

where s measures the difference in start times for first flight f in set B between pairs $(\dot{A}_1, B)$ and $(\dot{A}_2, B)$, and rs ($r = |B|$) is the corresponding minimum increase or maximum decrease in delay. Now, if $D \geq 0$, it is not necessary to optimize set B in the second (sibling) pair, as the optimal delay value of the second pair is atleast equal to optimal value of the first pair.

Case 2. If a particular flight f is not required to be at first position in every optimal solution of set B , then the minimum difference in start times with respect to all possible flights which can occupy the first position in set B needs to be determined. For this, calculate

$$s = (\dot{z}_2 - \dot{z}_1) + \min_{\hat{f} \in \hat{\mathcal{F}}} (\Delta_{i_2\hat{f}} - \Delta_{i_1\hat{f}}),$$

if $s \geq 0$, it can be inferred that the delay for set B in instance $(\dot{A}_2, B)$ is atleast the same as the delay for set B in instance $(\dot{A}_1, B)$. Moreover, if $\dot{d}_{\dot{z}_2} - \dot{d}_{\dot{z}_1} \geq 0$, the second pair can be fathomed since the optimum value of the second pair is greater than or equal to that of the first pair.

3.3.4 DS-ASP algorithm: Extension to Mixed-Traffic

In the scenario where FCFS sequence consists of both arrival and departure aircraft, and every aircraft has a CPS constraint with respect to its FCFS sequence position, DS-algorithm can be extended to the case of mixed traffic. There will be two key differences in algorithm when extending it from arrival only to mixed traffic: (i) first, larger number of instance-pairs $(\dot{A}, B)$ are generated as compared to arrival-only case, since triangular inequality ($\Delta_{ij} + \Delta_{jk} \geq \Delta_{ik}$ for any 3-tuple of flights (i, j, k)) with respect to the safe separation distance may not always be satisfied, and (ii) second, the cases of $\delta_1 > 0$ are treated, since likelihood of early landings for mixed-traffic is very high in practical range of γ (see Section 3.4 for more details).

Preparation of instance pairs

When dealing with the segregated case of only arriving traffic, as the triangular inequality with respect to the safe separation distance matrix is always satisfied, it is sufficient to maintain the separation of the first aircraft in set B from the

last aircraft in $\dot{A}$; however, it may violate the triangular inequality in case of mixed-traffic for the following solution instances:

Case 1: Of the last two flights i and j from set $\dot{A}$, the second last is a heavy arrival, followed by a departure of any aircraft type. That is then followed by a large arrival k (from set B).

Case 2: Of the last two flights i and j from set $\dot{A}$, the second last is a heavy arrival, followed by a departure of any aircraft type. That is then followed by a small arrival k (from set B).

In Case 1 (based on the given time-separation matrix (Table 1.1)), $\Delta_{ij} + \Delta_{jk} = 135$ seconds, whereas $\Delta_{ik} = 157$, thereby resulting in a violation. Similarly, a violation occurs in the latter case. Hence, in order to ensure that such violations do not occur, the original pair $(\dot{A}, B)$ consisting of a departure at last position in $\dot{A}$ is replaced with two new pairs along with the necessary *non-triangular* separation constraints. The first pair fixes the heavy aircraft at second last position in $\dot{A}$, and second pair does not allow any heavy aircraft at this position; there will be no change for flights in set B . To illustrate these cases, consider the example data from Section 3.1 ($\{H1, H2, H3, L4, L5, H6, S7, S8\}$), where now the flight at position 1 is a large arrival, flights at positions 4 is a departure, and the remaining flights remain same as before. In the case of $\dot{A} = \{L1, H2, H3, \dot{L}4\}$ and $B = \{L5, H6, S7, S8\}$, ($\dot{\bullet}$ denotes departure) the following additional instance pairs need to be generated to account for the resulting triangular inequality violations:

Pair 1: $\dot{A} = \{_, _, H3, \dot{L}4\}$, $B = \{L5, H6, S7, S8\}$

Pair 2: $\dot{A} = \{_, _, L1, \dot{L}4\}$, $B = \{L5, H6, S7, S8\}$

Solving the instance pairs

For some cases of realistic instances of mixed traffic, the likelihood of early landings in a feasible schedule of the following set B is very high (see Section 3.4.1 for more detail). This renders Algorithm 3.1 ineffective for these instances as δ_1 may take non-zero value while solving an instance-pair, as minimum rate of increase of delay of set B is $< |B|$ in early landings scenario (refer to Proposition 3.4). Moreover, unlike the scenario where all flights are late in set B , estimating the minimum rate of increase is not possible as different optimal sequences may have a different possible number of early and late aircraft in set

B . However, since time-windows are wider, delay still decreases at the rate $\leq |B|$ for any reduction from optimum makespan (calculated for $e = |B|$), say $\dot{z}_1$.

Fortunately, a solution can be found by determining the lower bound on the minimum rate of increase in delay, say $\bar{e}$, by looking at the sequence with least number of late aircraft. $|B|$ is then replaced with $\bar{e}$ in $Opt_{\dot{A}}$ and the new makespan corresponding to new Optima, say $\dot{z}_2$, is compared with $\dot{z}_1$. (Note that from Proposition 3.5, $\dot{z}_2 \geq \dot{z}_1$.) Finally, the following cases are investigated to conclude on optimality:

1. $\dot{z}_2 = \dot{z}_1$: This implies that the maximum rate of decrease in delay of leading set is equal to $\bar{e}$. Therefore, the overall delay cannot decrease as the minimum rate of increase in delay of the following set is $\bar{e}$, thereby, concluding optima. This case is critical to the success of this solution procedure, and, fortunately, is found to be very common in the computations, particularly when $\bar{e}$ is close to $|B|$.
2. $\dot{z}_2 > \dot{z}_1$: This case is tricky to handle. However, the lower bound on the total delay of the pair $(\dot{A}, B)$ in the process can be calculated. In the end, the incumbent (minimum of all delays related to the pairs for which optimality has been concluded) is compared with the minimum of delays of all pairs for which lower bound has been calculated; optimality for the given set is concluded when the former is less than or equal to the latter.

A formal statement of the proposed data-splitting algorithm is given in pseudo-code form next. In the algorithm, let $\dot{d}_m$ (resp. d_m) denote the delay of leading set (resp. following set) w.r.t makespan m .

Algorithm 3.2: Pseudo-code for mixed traffic

- 1: Set iteration counter $p \leftarrow 0$, $OptVal^{(p)} \leftarrow +\infty$, incumbent1, incumbent2 $\leftarrow OptVal^{(p)}$, $\mathcal{P} = \{\text{set of all pairs } (\dot{A}, B)\}$.
- 2: If $\mathcal{P} = \emptyset$, go to Step 11; else, set $p \leftarrow p + 1$, go to Step 3.
- 3: Arbitrarily select one instance pair $(\dot{A}_p, B_p)$ from $\mathcal{P}$.
- 4: Solve $Opt_{\dot{A}}$ for set $\dot{A}_p$ and $e = |B_p|$. Let $\dot{z}_1$ be the completion time and $\dot{d}_{\dot{z}_1}$ be the optimal delay value.
- 5: Compute lower bound on rate of increase of delay, say $\bar{e}$. If $\bar{e} < |B_p|$, go to Step 6; else, set $\dot{z}_2 = \dot{z}_1$, $\bar{e} = |B_p|$, go to Step 7.
- 6: Solve $Opt_{\dot{A}}$ for set $\dot{A}_p$ with $e = \bar{e}$. Let $\dot{z}_2$ be the completion time and $\dot{d}_{\dot{z}_2}$ be the optimal delay value.

- 7: Solve Opt_B for set B_p , with coefficient of δ_1 and δ_2 being $\bar{e}$ and $|B_p|$. Let $d_{\dot{z}_1}$ be the optimal delay. If $\delta_1 = 0$ and $\delta_2 = 0$ go to Step 8; else, print "Optimality can not be concluded".
- 8: If $\dot{z}_2 - \dot{z}_1 = 0$, go to Step 9; else, calculate lower bound on total delay as $LB = d_{\dot{z}_2} + d_{\dot{z}_1} + \bar{e}(\dot{z}_2 - \dot{z}_1)$, set $incumbent2 \leftarrow LB$ if $incumbent2 > LB$, update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\dot{A}_p, B_p\}$, go to Step 2.
- 9: Concatenate the optimal sequences; set $OptVal^{(p)} \leftarrow \sum_{i \in \dot{A}_p \cup B_p} (|t_i - T_i|)$.
- 10: If $OptVal^{(p)} < incumbent1$, set $incumbent1 \leftarrow OptVal^{(p)}$. Update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\dot{A}_p, B_p\}$. Go to Step 2.
- 11: If $incumbent1 \leq incumbent2$, return $incumbent1$ as the best solution; else, print "Optimality can not be concluded". STOP.

1: **function** CALCULATE-LOWER-BOUND-RATE()

2: Solve,

$$\min \left\{ \sum_{i \in B_p} u_i : (3.1) - (3.6), t_i \geq t_l + \Delta_{li}, Mu_i \geq t_i - T_i \right\}, \text{ where } u_i \text{ is an indicator binary variable constrained by } Mu_i \geq t_i - T_i \text{ (} M = L_i - T_i \text{) to take the value 1, if flight } i \text{ is late.}$$

3: Calculate $\bar{e} = 2 \sum_{i \in B_p} u_i - |B_p|$. (Minimum rate of increase = minimum number of late flights – rest.)

4: **return** $\bar{e}$

5: **end function**

Next, the proof is given for the validity of the statement $\dot{z}_2 \geq \dot{z}_1$, and the expression to calculate lower bound on delay is developed.

Proposition 3.5. $\bar{e}$ less than e may increase the optimal makespan of the leading set from $\dot{z}_1$.

PROOF. This is proved by contradiction. Let $\bar{e}$ reduce the makespan by δ . Now, since optima occurs at $\dot{z} - \delta$,

$$\begin{aligned} d_{\dot{z}-\delta} + \bar{e}(\dot{z} - \delta) &\leq d_{\dot{z}} + \bar{e}\dot{z} \\ \Rightarrow d_{\dot{z}-\delta} - d_{\dot{z}} &\leq \bar{e}\delta. \text{ (on rearranging)} \end{aligned}$$

It is contradictory, since from Lemma 3.1, $d_{\dot{z}-\delta} - d_{\dot{z}} \geq e\delta$. Hence, the proof. $\square$

Proposition 3.6. $d_{\dot{z}_2} + d_{\dot{z}_1} + \bar{e}(\dot{z}_2 - \dot{z}_1)$ is the lower bound on overall delay for a particular pair.

PROOF: With increase in optimal makespan of $\dot{A}$ from $\dot{z}_1$ to $\dot{z}_2$, delay in set $\dot{A}$ decreases at the rate greater than or equal to $\bar{e}$, whereas delay in set B increases at the rate $\geq \bar{e}$. Therefore, a lower bound on optimal delay of this pair can be calculated as the summation of delay of set $\dot{A}$ at $\dot{z}_2$, i.e. $\dot{d}_{\dot{z}_2}$, and minimum delay of set B at $\dot{z}_2$, i.e. $d_{\dot{z}_1} + \bar{e}(\dot{z}_2 - \dot{z}_1)$. Hence, the proof. $\square$

REMARK 3.1: The enhancement of DS-ASP (refer Section 3.3.3) can also be applied to mixed traffic in case the separation of flights in the following set is needed only from the last flight in the leading set. $r = \bar{e}$ is set when $s \geq 0$, else $r = |B|$. $\square$

3.4 Computational Results

The proposed data-splitting algorithm is tested on several randomly generated instances comprising of thirty or forty aircraft, under both arrival and mixed-traffic mode. The aircraft mixture for these instances is set as 40% Heavy, 40% Large, 20% Small aircraft. The target times of aircraft are randomly generated, assuming that each aircraft appears *Uniformly* once every γ seconds. We also consider a wide time-windows scenario where a flight can only arrive (or depart) up to 60 seconds earlier and no later than 1800 seconds past its scheduled target time (see Section 3.1). All of the computations are performed on a Windows machine, equipped with an Intel Xeon CPU E5-1630v3 3.70GHz 32GB RAM processor, using MATLAB and Gurobi 6.0.0 (www.gurobi.com) as the underlying MIP-solver. The time-limit (TL) for computations is set at 3600 seconds.

Multiple randomly generated instances for both arrival-only and mixed-traffic (60% arrival and 40% departure) scenarios are solved by setting $\gamma = 60$ seconds and $k \in \{1, 2, 3\}$. As compared to the original ASP formulation (3.1)–(3.6) (including variable fixing strategies), the proposed DS-ASP algorithm results in over 90% reduction in the computational time required to solve all the randomly generated instances (Table 3.2). Note that the average is taken over ten randomly generated instances for each k , and the range corresponds to the difference of minimum and maximum solution time obtained over ten instances. Furthermore, from Table B.1 (Appendix B), a significant reduction in B&B nodes explored by the solver is observed when both the methods are compared. This tremendous reduction in computational effort can be attributed to three reasons: (i) a

significant reduction in search space as discussed in the combinatorial exposition (Appendix B); (ii) a difference in solver behavior when solving large-scale versus small-scale instances, with the solver being able to close the gap more efficiently for smaller instances (verified by the consistency in solution time), and (iii) time taken to solve the LP-relaxation at each node is significantly lower in case of smaller instances.

Table 3.2 Computational time comparison between DS-ASP and ASP where superscript a denotes arrival and m denotes mixed traffic

	Time (s)						% Reduction
$k \rightarrow$	1		2		3		
	DS-ASP	ASP	DS-ASP	ASP	DS-ASP	ASP	
F	Avg. (Range)	Avg. (Range)	Avg. (Range)	Avg. (Range)	Avg. (Range)	Avg. (Range)	
30^a	0.62 (0.39-0.85)	0.7 (0.47-1.03)	2.3 (1.48-3.35)	29.14 (5.36-113.6)	7 (4.1-10.81)	>1174.5 (129.2->3600)	>99.1
40^a	4.49 (2.02-6.11)	2.58 (0.55-6.59)	12.67 (3.31-22.9)	>1012.59 (14.39->3600)	52.35 (7.02-123.92)	>3293.72 (536.43->3600)	>98.4
30^m	1.09 (0.51-1.61)	0.66 (0.3-1.31)	5.72 (3.1-8.4)	118.81 (17.47-262.08)	24.94 (15.42-36.26)	>1395.23 (84.67->3600)	>97.9
40^m	6.2 (3.01-9.21)	2.11 (1.19-3.08)	28.67 (17.55-41.21)	>2587.52 (790.49->3600)	439.38 (196.61-1357.05)	> 3600 (>3600->3600)	>92.3

3.4.1 Effect of γ

The inter-arrival time, γ , is found to be a very important factor affecting the success of DS-ASP as it may cause both δ_1 and δ_2 to take on non-zero values, thereby affecting the convergence of the algorithm to optimality.

1. Effect on δ_1 : When γ is in range of average optimal separation of the aircraft, δ_1 can take non-zero; the following set may observe some early landings, in this case, thereby causing the minimum rate of increase of delay to be lower than estimated rate $|B|$.

In case of realistic instances of arrival traffic (see Section 3.3), we infer that this deviation is not possible as the most conservative estimate on average required separation time is found to be 79 seconds, whereas γ is around 60 seconds. (To estimate this we take aircraft mixture of 60% large and 40% heavy aircraft, instance size of 30, relax time-windows and CPS constraints, and then use the group clustering (refer Appendix A for this method)—for all other realistic instances, the estimate will be higher.) Moreover, in our computations which include CPS and time windows, it is always observed that the average separation time to be more than 85 seconds.

In case of mixed traffic, the deviation from $|B|$ is likely, particularly for $k = 3$, as required average separation may become close to 60 seconds for some combination of traffic-mixture and instance size. So, in this case, Algorithm 3.2 can be used which, though, may not converge to optimality for all the instances. Next, the effect of γ is discussed on success rate in case of mixed traffic, the aircraft and traffic mixture is same as aforementioned.

Figure 3.3a illustrates the success rate of the DS-ASP algorithm for varying γ and Figure 3.3b illustrates the algorithmic efficiency of DS-ASP in directly solving the MIP model for the mixed-traffic scenario for $k = 3$ and 30 flights. It can be observed that for $\gamma = 60$, the overall success rate of the algorithm is 90% (it is found to be 95% for $k \leq 2$), which improves to 100% for $\gamma = 55$. Lowering the γ moves the minimum rate of increase of delay closer to $|B|$ (or almost $|B|$) for all pairs of an instance resulting in higher effectiveness of Algorithm 3.2 as sensitivity of optimal makespan for $Opt_{\bar{A}}$ is observed to be very limited for a small change in $|B|$. Furthermore, the increasing success ratio with lowering γ turns out to be very important as the computational time taken by the MIP solver increases exponentially (> 810 seconds for $\gamma = 55$) with a decrease in γ —DS-ASP always consumes significantly lower resources (12 seconds for $\gamma = 55$) and is relatively consistent.

The airport managers should always use DS-ASP first and then resort to other methods only when it fails to converge. DS-ASP's cost of computation is very low, and for the cases where it fails to converge, it is even lower.

Fig. 3.3 Effect of γ on success and efficiency of algorithm in case of mixed traffic

2. Effect on δ_2 : When γ is too low in comparison to the average optimal separation of the aircraft, the better sequence can result as the sequence is pulled in (as a result of a decrease in $\hat{z}$), thereby causing δ_2 to take non-zero value. The following analysis is presented to find the minimum γ above which δ_2 could

remain zero. In an optimal sequence of the following set, the maximum time at which a flight i can land/depart can be approximately given as $t_i = (i + k)a$, where i is the position in FCFS sequence, and a is the average optimal separation. Further, the permissible latest time by which this flight must land/depart is approximate $L_i + \gamma i$. Next, if all the flights in the following set are well within their permissible time-windows, then decreasing t_i (caused by a decrease in makespan of the leading set) cannot result in a better sequence since a better sequence must be currently feasible too. Mathematically, for δ_2 to remain zero,

$$\begin{aligned} L_i + \gamma i &\geq (i + k)a \quad \forall i \in \mathcal{F} \Rightarrow i(a - \gamma) + ka \leq L_i \quad \forall i \in \mathcal{F} \\ \Rightarrow \max_{i \in \mathcal{F}} (i(a - \gamma) + ka) &\leq L \quad (L = \min_{i \in \mathcal{F}} L_i) \\ \Rightarrow F(a - \gamma) + ka &\leq L \Rightarrow \gamma \geq \frac{a(F+k)-L}{F} \end{aligned}$$

Using the above formula, the minimum γ for arrival traffic and mixed traffic for the largest possible instance size can be given as:

1. Arrival: ($F = 40, a = 90, k = 3, L = 1800$): 51.8
2. Mixed: ($F = 50, a = 70, k = 3, L = 1800$): 38.2

Note that reducing the γ further may render the problem infeasible for the given instance size.

3.4.2 Parallelization

Furthermore, to illustrate the benefits of parallelization, Figure 3.4 and 3.5 plot the delay versus the solution time corresponding to each instance pair, for sample problems of size 30 and 40, respectively. It can be observed that with parallelization, the optimal solution for a particular problem of size 30 can be reached in 1.01 seconds, and for size 40, it can be reached in 2.91 seconds, whereas it takes around 5.58 second and 34.39 seconds, respectively, when the problems are solved sequentially. Furthermore, the implementation of this parallelization scheme is very simple as it only requires solving the instance pairs on different processors and comparing their solution at the end.

Fig. 3.4 Optimal delay and time for each instance pair for an instance of size of 30

Fig. 3.5 Optimal delay and time for each instance pair for an instance of size of 40

3.5 Summary

In this chapter, the aircraft sequencing problem (ASP) on a single runway is considered under both segregated and mixed traffic conditions, wherein an optimal sequence and schedule for a set of flights is determined, respecting safety, equity, and time-windows constraints, while minimizing the total delay in the system. Recognizing the inability of directly using a 0–1 MIP approach to provide real-time optimal solutions for large-scale realistic instances of this problem, a novel *data-splitting algorithm* is proposed, which is believed to be the first tractable solution for the problem under consideration. The data-splitting algorithm decomposes the original flight data-set into several potential pairs of leading and following aircraft sets and then determines the optimal solution by application of the 0–1 MIP on these smaller data sets. The computational experiment clearly demonstrates that optimality can be concluded in real-time for nearly all practical test instances.

There is a huge practical implication of this study as the proposed algorithm can handle the large scale realistic instances in peak traffic condition. In this case,

the inter-arrival times are small (i.e., the airport is busier) causing our algorithm to converge at a very high rate. Moreover the times required are about the same, as the rate of convergence changes. On the other hand, directly solving the MIP at busy airport takes much more time.

In the next chapter, ASP on a single runway with the objective of maximizing throughput is investigated. Further, observing the absence of tractable LP-based branch-and-cut approaches as well as the inability of dynamic programming—the existing state-of-the-art for this objective—in tractably handling the certain problem scenarios, DS-ASP is tailored to solve this problem. Additionally, a new algorithmic enhancement routine and framework of multiple splits is proposed.

Chapter 4

Aircraft Sequencing on a Single Runway to Maximize Throughput: A Data-Splitting Approach

4.1 Introduction

This chapter formulates and solves the version of ASP as defined in Balakrishnan and Chandran (2010) that exactly optimizes the sequence of flights on a single runway while incorporating all practical constraints including CPS, wide time-windows, precedence restrictions, early landings/departures—and the objective of maximizing throughput (or equivalently minimizing the completion time). Being a potential candidate to be deployed in many arrival-departure management systems, this problem has great practical implications. Being an NP-hard problem, obtaining optimal solutions in real-time for large scale realistic instances of this problem is a challenge.

However, exploiting one or the other problem characteristics of the ASP, *dynamic programming* (DP) provides an efficient solution for a specific set of scenarios. Assuming that the aircraft from the same category are identical, all aircraft are ready to land together, and ignoring the time-window considerations, Psaraftis (1980) provides the first efficient dynamic programming implementation building on the fact that the number of aircraft categories are limited. His algorithm is exponential in number of categories and polynomial in number of aircraft, and can provide a *tractable* solution with or without CPS requirements,

under the aforementioned scenario. However, considering that the above formulation cannot accommodate time-windows and precedence restrictions, the two important practical constraints, another DP algorithm is offered by Balakrishnan and Chandran (2010) whose implementation is exponential in k (MPS parameter) and polynomial in n (instance size). Even though, their formulation includes most of the practical constraints and is tractable under the segregated mode of operations for values of k up to 3, it may face the curse of dimensionality in mixed-mode of operations as violation of triangular inequality for some of the aircraft triplet requires to include time in the state function. Furthermore, increasing the value of k above 3 renders it intractable. In addition, of all possible optima, DP approach is unable to provide a specific minimum makespan schedule that leads to associated minimum delay¹; this requires minimizing delay to be included in the objective function, and its inclusion renders DP computationally prohibitive. Table 4.1 compares the minimum delay (averaged over ten randomly generated instances) obtained with and without it in the objective function; notably, without any loss of makespan, around 2 minutes of delay is saved in case of former. The above discussion necessitates the development of a tractable approach which is more *robust*.

Table 4.1 Comparison of delay with and without minimizing delay as an objective function component

k	With Delay	Without Delay
1	14240	14297
2	12935	13084
3	12049	12151

Next, ASP (the version under consideration) is formulated as a 0–1 *mixed-integer program* (**MIP**), an adaptation of Beasley et al. (2000)’s model, enhanced with the inclusion of the CPS constraints. Recognizing that a more robust approach over current state-of-the-art is needed, this chapter proposes an extension to data-splitting algorithm (DS-ASP) discussed in Chapter 3, that optimizes the flight sequences by a repeated application of the 0–1 MIP on smaller data sets while ensuring the global optimality. Another important and intrinsic feature of the proposed data-splitting framework is that it is pleasingly parallelizable, making it amenable to rapidly scalable computations.

¹There can be several optima with the same optimal makespan but different values of delay.

There are four significant contributions of this work:

1. The proposed data-splitting algorithm is more robust than state-of-the-art DP approach for the large scale realistic instances.
2. To the best of our knowledge, this is the first LP-based branch-and-cut approach to provide a tractable solution of the problem under consideration. The branch-and-cut solutions for ASP provided by Beasley et al. (2000), Ghoniem et al. (2015) are intractable for $k \geq 2$. (Note that CPS is incorporated separately as the original formulations have not considered it.)
3. It is a potential candidate algorithm to be included in arrival-departure management systems used in practice since LP-based branch-and-cut algorithm is able to account for most of the operational constraints.
4. It motivates the use of DS-ASP for various scheduling problems to obtain high quality solution in real time, even where CPS constraints may not be present.

The remainder of this chapter is organized as follows. In Section 4.2, the 0–1 MIP formulation for the ASP along with some preliminary computations are presented that serve to highlight the inherent difficulty in solving ASP via a straightforward MIP approach. Next, in Section 4.3, the details of the data-splitting algorithm, and its pseudo-code are described, along with algorithmic insights and routines to enhance the computational performance. Then, Section 4.4 presents detailed computational results and analysis pertaining to large-scale (realistic) instances, and finally, Section 4.5 summarizes the contributions of this work and suggests extensions for future research.

4.2 Optimization Problem

Next, the ASP is formulated as a 0–1 mixed integer program, where the index sets and parameters are as defined below.

Description of Index Sets and Parameters

- $\mathcal{F}$: Set of all arriving and departing flights.
- E_i : Earliest time of arrival (departure) of aircraft i .
- L_i : Latest time of arrival (departure) of aircraft i , where $L_i > E_i$.
- T_i : Target time of arrival (departure) of aircraft i , where $E_i \leq T_i \leq L_i$.

- Δ_{ij} : Mandatory minimum safety separation (in seconds) enforced at the runway threshold, if flight i is ahead of flight j (Table 1.2).
- $seq_S(i)$: Position of flight i in any feasible sequence S .
- k : Maximum position shifting (MPS) parameter.

Decision Variables

Decision Variables:

$$x_{ij} = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}$$

t_i = Scheduled time of arrival/departure of flight i

z = Runway makespan.

$$\text{ASP :} \quad \text{Minimize} \quad z \quad (4.1)$$

$$\text{subject to:} \quad x_{ij} + x_{ji} = 1, \quad \forall i < j, (i, j) \in \mathcal{F} \quad (4.2)$$

$$t_j \geq t_i + \Delta_{ij} - M(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{F} \quad (4.3)$$

$$i - k \leq (F - \sum_{j \in \mathcal{F}, i \neq j} x_{ij}) \leq i + k, \quad \forall i \in \mathcal{F} \quad (4.4)$$

$$z \geq t_i \quad \forall i \in \mathcal{F} \quad (4.5)$$

$$E_i \leq t_i \leq L_i, \quad \forall i \in \mathcal{F} \quad (4.6)$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in \mathcal{F}; \quad t_i \geq 0, \quad z \geq 0, \quad \forall i \in \mathcal{F}. \quad (4.7)$$

In the above formulation, the objective function (4.1) seeks to minimize the makespan (completion time); constraint (4.2) enforces the order precedence relationship between flights i and j ; constraint (4.3) ensures the time-based separation requirements between flights at the runway (as mandated by the FAA

for various flight classes (see Table 1.2), where $M \equiv (L_i + \Delta_{ij} - E_j)$; constraint (4.4) imposes the CPS constraint that an aircraft cannot be shifted by more than ' k ' positions from its initial (FCFS) position, where for the sake of notational convenience, let $seq_{FCFS}(i) \equiv i, \forall i \in \mathcal{F}$, i.e., aircraft are labeled based on their FCFS priority order, and $F \equiv |\mathcal{F}|$; constraint (4.5) ensures that makespan z must be greater or equal to scheduled time of last aircraft; constraint (4.6) maintains the scheduled time of arrival (departure) at the runway to be between the earliest and latest times for each aircraft; and finally constraint (4.7) imposes binary and non-negativity restrictions on the x -, t -, and z variables, respectively.

Note that ASP formulation described above can be further improved by using variable-fixing strategies based on CPS and ELW rule presented in Chapter 3. Further, precedence constraints are not explicitly included in the formulation as they are airport specific and can be easily accounted for by fixing the variables based on input data. Note that, if an airport administers precedence restrictions, then ELW rule should not be implemented as it may lead to sub-optimality.

To gain an understanding of the computational difficulties involved in solving the ASP, the optimization model, given by (4.1)–(4.7) and enhanced with the aforementioned variable fixing strategies, is tested on several randomly generated large-scale, realistic, instances. From the results (Table 4.2), it can clearly be seen that the computational time increases exponentially and the MIP model fails to converge to optimality in a preset time-limit, particularly for cases where the maximum position shifting parameter $k > 2$.

Table 4.2 Computational times based on a direct implementation of Problem ASP on large-scale instances, time-limit is set as 3600 seconds

Instance	# of flights	Mode of Operation	MPS (k)	
			2	3
1	30	Mixed	61	614
2	40	Arrival	551	> 3600
3	40	Arrival	233	> 3600
4	40	Arrival	171	> 3600
5	40	Mixed	2241	> 3600

4.3 Implementation of DS-ASP

Data-splitting algorithm (DS-ASP) discussed in detail in previous chapter is implemented in two stages. The first stage is instance generation, in which the

original flight data-set is divided into several possible pairs of disjoint subsets of leading and following aircraft. The second stage is the solution stage, in which each resulting subset pair is independently solved to determine its optimal solution. The obtained solutions are finally compared to get the overall optimum.

Even though the basic framework of DS-ASP stays the same, the procedure varies from application to application. In the case of minimizing makespan of a single runway, the instance generation scheme is the same as in minimizing delay on a single runway (see Chapter 3), however, the solution procedure would be different as described in Section 4.2. Moreover, an effective algorithmic enhancement routine and multiple splitting of the data-set (in order to handle large sized instances) are new features of DS-ASP discussed below; these routines are designed to exploit a characteristic of minimizing makespan function that it is a *separable* objective function (Psaraftis 1978), i.e., the optimal makespan of a given sequence can be determined from the optimal makespan of its subsequences.

4.3.1 Solving the instance pairs

As aforementioned, since the makespan objective function is separable, minimizing the makespan of leading and following sets independently will minimize the makespan of the given pair. Now, the optimization problems for sets $\dot{A}$ and B which will be fed to the algorithm can be summarized as:

$$Opt_{\dot{A}} \equiv \min \{z, \text{ s.t. (4.2) – (4.7), } \forall i \in \dot{A}\}.$$

$Opt_B \equiv \min \{z, \text{ s.t. (4.2) – (4.7), } t_i \geq t_{\dot{l}} + \Delta_{\dot{l}i}, \forall i \in B\}$, where $\dot{l}$ is the last aircraft in $\dot{A}$ from which aircraft in the following set have to maintain the minimum separation. In case of mixed-traffic, extra set of constraints ($t_i \geq t_{\dot{l}} + \Delta_{\dot{l}i}, \forall i \in B$) for separation from second last aircraft must also be included when a heavy arrival is fixed at the second last position in $\dot{A}$.

The pseudo-code of the data-splitting algorithm is listed next.

Algorithm 4.1: Pseudo-code for single-split

- 1: Set incumbent $\leftarrow +\infty$, $\Pi \leftarrow \emptyset$, and $\mathcal{P} = \{\text{set of all pairs } (\dot{A}, B)\}$.
- 2: If $\mathcal{P} = \emptyset$, stop; return incumbent and Π as the best solution. Else, go to Step 3.
- 3: Arbitrarily select one instance pair $(\dot{A}, B)$ from $\mathcal{P}$.

-
- 4: Solve $Opt_{\dot{A}}$. Let $\dot{l}$ and $t_{\dot{l}}$ be the last scheduled aircraft and its time of arrival/departure.
 - 5: Solve Opt_B . Let $\hat{z}$ be the optimal solution.
 - 6: Update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\dot{A}, B\}$. If $\hat{z} < \text{incumbent}$, set $\text{incumbent} \leftarrow \hat{z}$ and update Π with the optimal solution of pair $(\dot{A}, B)$; go to Step 2.
-

4.3.2 Algorithmic Enhancements by fathoming “sibling” instance pairs

Consider two instance pairs $(\dot{A}_1, B)$ and $(\dot{A}_2, B)$, resulting from the same parent pair (A, B) . Let $\hat{\mathcal{F}}$ represent the set of flights in set B between positions $(m+1)$ to $(m+k+1)$, and let $(\dot{z}_1, \dot{l}_1)$ and $(\dot{z}_2, \dot{l}_2)$ denote the completion time, and last flight in the optimal solutions for set $\dot{A}_1$ and $\dot{A}_2$, respectively. Furthermore, suppose that Set B has been solved to optimality in pair $(\dot{A}_1, B)$, with flight $f \in \hat{\mathcal{F}}$ placed at the first position in the optimal solution of set B , and $\hat{z}$ is the optimal throughput of pair $(\dot{A}_1, B)$. Now, consider the following cases:

Case 1. Suppose that flight f is bound to the first position in any optimal solution of set B . Calculate,

$$s = (\dot{z}_2 - \dot{z}_1) + (\Delta_{\dot{l}_2 f} - \Delta_{\dot{l}_1 f})$$

where s measures the difference in start times for the first flight f in set B between pairs $(\dot{A}_1, B)$ and $(\dot{A}_2, B)$.

Case 2. If flight f is not bound to first position in any optimal solution of set B , then it is needed to determine the minimum difference in the start times with respect to all the flights which can come at the first position in set B . Towards this end, calculate,

$$s = (\dot{z}_2 - \dot{z}_1) + \min_{\bar{f} \in \hat{\mathcal{F}}} (\Delta_{\dot{l}_2 \bar{f}} - \Delta_{\dot{l}_1 \bar{f}}),$$

Now, consider the following conditions on s :

1. $s \geq 0$. Under this scenario, optimizing set B in the second (sibling) pair can be foregone, since the optimal value of the second pair is at least equal to the optimal value of the first pair.
2. $s < 0$. Careful examination of optimal makespan $\hat{z}$ reveals that

$$\hat{z} = \dot{z}_1 + \Delta_{\dot{l}_1 f} + \bar{z}_f.$$

where $\bar{z}_f$ is the makespan length corresponding to set B with flight f at the first position. Since, flight f corresponds to the optimal makespan, clearly for other

flight $\bar{f}$ which can occupy first position in set B ,

$$\begin{aligned}\hat{z} \leq \dot{z}_1 + \Delta_{i_1 \bar{f}} + \bar{z}_{\bar{f}} &\Rightarrow \bar{z}_{\bar{f}} \geq \hat{z} - \dot{z}_1 - \Delta_{i_1 \bar{f}}. \\ &\Rightarrow \bar{z}_{\bar{f}} \geq \bar{z}_f + \Delta_{i_1 f} - \Delta_{i_1 \bar{f}}. \\ &\Rightarrow \bar{z}_{\bar{f}}^{\min} = \bar{z}_f + \Delta_{i_1 f} - \Delta_{i_1 \bar{f}}.\end{aligned}$$

Therefore, the lower bound on optimal makespan of pair $(\dot{A}_2, B)$ is given as,

$$\min_{\bar{f} \in \mathcal{F}} (\dot{z}_2 + \Delta_{i_2 \bar{f}} + \bar{z}_{\bar{f}}^{\min})$$

These lower bounds are compared, at the end, with minimum of all the makespan for which optimality has been concluded. If the former is smaller than the latter, optimization is run for the corresponding pair, else, it is aborted. Note that this algorithmic enhancement is only applicable in the scenario where separation of flights in the following set is only needed from the last flight in the leading set.

The pseudo-code of the enhanced data-splitting algorithm is listed next.

Algorithm 4.2: Enhanced data-splitting algorithm

- 1: Set incumbent $\leftarrow +\infty$, $\Pi \leftarrow \emptyset$, and $\mathcal{S} = \{\text{set of all pairs } (A, B)\}$.
 - 2: If $\mathcal{S} = \emptyset$, go to final Step; else, go to Step 3.
 - 3: Arbitrarily select one instance pair (A, B) from $\mathcal{S}$, set $flag \leftarrow 0$. Let $\dot{\mathcal{S}} = \{\text{set of all pairs } (\dot{A}, B)\}$ corresponding to (A, B) .
 - 4: If $\dot{\mathcal{S}} = \emptyset$, update $\mathcal{S} \leftarrow \mathcal{S} \setminus \{A, B\}$, go to Step 2. Else, arbitrarily select one instance pair $(\dot{A}, B)$ from $\dot{\mathcal{S}}$.
 - 5: Solve $Opt_{\dot{A}}$. Let $\dot{l}$ and $t_{\dot{l}}$ be the last scheduled aircraft and its arrival/departure time. If $flag = 0$ or separation is also needed from second last aircraft in $\dot{A}$, set $flag = 1$ if only former condition is satisfied, go to Step 6; else, go to Step 7.
 - 6: Solve Opt_B . Let $\hat{z}$ be the optimal solution. Update $\dot{\mathcal{S}} \leftarrow \dot{\mathcal{S}} \setminus \{\dot{A}, B\}$. If $\hat{z} < \text{incumbent}$, set incumbent $\leftarrow \hat{z}$ and update Π with the optimal solution of pair $(\dot{A}, B)$, go to Step 4.
 - 7: Calculate s . If $s \geq 0$, set $\hat{z} \leftarrow +\infty$; else, calculate the lower bound and save it in memory. Update $\dot{\mathcal{S}} \leftarrow \dot{\mathcal{S}} \setminus \{\dot{A}, B\}$, go to Step 4.
 - 8: For each saved entry: If lower bound is $\geq \text{incumbent}$, discard it; else solve Opt_B and update incumbent and Π in case $\hat{z} < \text{incumbent}$. Return incumbent and Π as the best solution.
-

4.3.3 Multiple Splits

When handling very large sized instances, such as $n = 60$, splitting the data set once may not lead to computationally efficient solutions as each resulting set is still quite large. One way to address this is multiple splitting wherein a data-set is split into groups of more than two subsets; however, the challenge then is to obtain optimality through independent solutions of several subsets. Fortunately, the independent solutions with the objective of minimizing makespan automatically lead to optima as minimizing the makespan of a preceding set also maximizes the search space of the following set.

In case of multiple splits data-splitting algorithm is applied recursively: first a data-set is split into two sets resulting in several instance pairs; subsequently each subset of a pair is split again if its cardinality is more than a user specified minimum, say ϕ . Note that ϕ is very important parameter as it controls the computational efficiency of the algorithm, solving a very small set of flight by using multiple splitting can be computationally expensive; on the other hand, solving a very large set of flight with only single split can also be computationally expensive—sometimes even intractable. A comparatively higher degree of parallelization can be achieved with multiple splits, thus solving any instance, however large, in real time. For the sake of comprehension, the algorithmic enhancements of Section 4.3 are ignored in the listing of pseudo-code of the multiple splitting algorithm that follows.

Algorithm 4.3: Pseudo-Code for multiple-split

Initialize: $\mathcal{S} = \mathcal{F}, t_i = 0, \dot{l} | \Delta s_{i\dot{l}} = 0 \forall i \in \mathcal{F}$. Optimal Decision = DATA-SPLITTING ($\mathcal{S}, t_i, \dot{l}$)

function DATA-SPLITTING ($\mathcal{S}, t_i, \dot{l}$)

- 1: Set incumbent $\leftarrow +\infty$, $\Pi \leftarrow \emptyset$, and split $\mathcal{S}$ into $\mathcal{P} = \{\text{set of all pairs } (\dot{A}, B)\}$ with instance sizes n_1 and n_2 .
- 2: If $\mathcal{P} = \emptyset$, stop; return incumbent as the best solution. Else, go to Step 3.
- 3: Arbitrarily select one instance pair $(\dot{A}, B)$ from $\mathcal{P}$.
- 4: If $n_1 \geq \phi$, call DATA-SPLITTING ($\dot{A}, t_i, \dot{l}$); else, solve $Opt_{\dot{A}}$ with extra constraints $t_i \geq t_j + \Delta_{i\dot{l}} \forall i \in \dot{A}$. Let $\dot{l}$ and t_i be the last scheduled aircraft and its time of arrival/departure.

-
- 5: If $n_2 \geq \phi$, call DATA-SPLITTING (B, t_j, l) ; else, solve Opt_B . Let $\hat{z}$ be the optimal solution.
 - 6: Update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\dot{A}, B\}$. If $\hat{z} < \text{incumbent}$, set $\text{incumbent} \leftarrow \hat{z}$ and update Π with the optimal solution of pair $(\dot{A}, B)$; go to Step 2.
-

4.4 Computational Results

In this section, with a prototype implementation, the computational performance of DS-ASP is showcased for various large-sized realistic instances under various modes of operations. Furthermore, the prowess of DS-ASP over ASP and DP is demonstrated in real-time handling of instances under various scenarios.

The experimental design is based on a realistic scenario as follows: (i) aircraft are randomly generated with mixture of 40% Heavy + 40% Large + 20% Small; (ii) considering the scheduling horizon of one hour and peak traffic conditions, instance sizes up to 40 are chosen for all the computational testing and comparisons, while instance size of 60 is chosen to demonstrate the benefit of multiple split; (iii) target times are randomly generated, assuming that each aircraft probabilistically appears *uniformly* once every γ seconds where γ is set to 50 or 60 seconds; (iv) for mixed mode scenarios, 50% are arrivals and 50% departures; and (v) earliest time of arrival for each flight is set to be 60 seconds before the target times as it is most economic for the arriving flight (Neuman and Erzberger (1991)), and latest time is set to be 1800 seconds past its target time due to fuel restrictions limit. The computational experiments are run on a Windows machine, equipped with a Intel(R) Xeon(R) CPU E5-1630 3.70GHz 32GB RAM processor, using MATLAB in conjunction with GuRoBi as the underlying MIP-solver. The time-limit (TL) is set as 3600 seconds, each computation represents the time in seconds averaged over ten randomly generated instances, and the range corresponds to the difference of minimum and maximum solution time obtained over ten instances. In the following discussion, the computational results are described for arrival traffic first, followed by mixed traffic and finally the scheme to use DS-ASP in scenarios without CPS restrictions is presented.

4.4.1 Arrival Traffic Only

Computations recorded in Table 4.3 compare the performance of DS-ASP on arrival traffic with DP and directly solving ASP (4.1)-(4.7) (including variable fixing strategies). Two key observations can be made from Table 4.3: (i) DS-ASP outperforms ASP for any k ; (ii) DP outperforms DS-ASP for $k \leq 2$, but not for $k > 2$; DP faces dramatic increase in computational time for $k = 4$. The reasons behind the DS-ASP's superior performance over ASP are: (i) a significant reduction in search space; (ii) the solver closing the gap more efficiently for smaller instances (verified by the consistency in solution time); and (iii) time taken to solve the LP-relaxation at each node is significantly lower in case of smaller instances. With regards to DP, the smaller state space in case of $k = 1$ or $k = 2$ renders DP remarkably efficient; however, $k > 2$ leads to DP's space explosion, and is outperformed by DS-ASP. The computational performance of DS-ASP is further enhanced (Table 4.4) through a simple parallel implementation.

Table 4.3 Computational time comparison for DS-ASP, ASP, and DP (– indicates DP is better)

$F \rightarrow$	30			40			% Savings (vs. DP)	% Savings (vs. ASP)
k	DS-ASP	DP	ASP	DS-ASP	DP	ASP		
1	0.3 (0.17-0.44)	0.01 (0.01-0.01)	0.055 (0.41-0.75)	0.82 (0.3-1.12)	0.01 (0.01-0.01)	2.6 (2.16-2.92)	–	67.2
2	0.86 (0.58-1.34)	0.34 (0.31-0.39)	11.2 (3.29-36.22)	3.28 (1.80-4.58)	0.43 (0.36-0.47)	366 (171.36-616)	–	99
3	1.8 (1.16-2.71)	35 (14.7-59.61)	77.6 (21.62-236.33)	9.71 (4.35-12.78)	43.7 (31.83-49.61)	> 3600 (> 3600-> 3600)	85.4	> 99.7
4	4.08 (1.89-2.15)	> 3600 (> 3600-> 3600)	360 (47.7-1309.05)	47.4 (14.4-81.2)	> 3600 (> 3600-> 3600)	> 3600 (> 3600-> 3600)	> 99.2	> 98.7

Table 4.4 Computational time comparison of DS-ASP, its parallel implementation, and DP

$F \rightarrow$	30			40		
k	DS-ASP	Parallel	DP	DS-ASP	Parallel	DP
1	0.5	0.2	0.01	0.76	0.21	0.01
2	1.8	0.38	0.34	3.45	0.95	0.25
3	3.67	0.69	35	12.4	4	58.6

Table 4.5 illustrates the computational benefits of multiple splits. For instance size 60, first set of computations are performed by splitting it into two equal data-sets of size 30, and second set by splitting it into four data-sets of size 15 each. Two important observations can be made from the results of Table 4.5. Firstly, a solution can be obtained in real time by multiple splits. Secondly, the

56 Aircraft Sequencing on a Single Runway to Maximize Throughput: A...

time savings are remarkable over single split, e.g. over 98% for instance size of 60, making DS-ASP even better than DP for $k = 3$.

Table 4.5 Computational time comparison of DS-ASP with one and two splits vs DP

$F \rightarrow$	40			60		
k	1-Split	2-Split	DP	1-Split	2-Split	DP
1	0.7	0.6	0.01	9.2	2.3	0.02
2	3.36	2.7	0.43	102	12.2	0.58
3	12.1	10	43.7	3100	51.2	99

4.4.2 Mixed Traffic

Table 4.6 shows the time taken by DS-ASP in comparison to ASP in solving instances of mixed traffic for value of k up to 3. Comparison of the performance with DP is not done in this case as DP discretizes the time-interval in handling mixed traffic, whereas DS-ASP handles continuous time. In addition, even after discretization, DP may have to evaluate a rather large number of states in case of mixed traffic, and is expected to perform rather poorly (the computations for mixed traffic scenario is not presented in Balakrishnan and Chandran (2010)). Notably, ASP is efficient in handling mixed-traffic scenario when $\gamma \geq 60$. The optimal average separation time is close to 60 seconds in case of mixed traffic, therefore $\gamma \geq 60$ provides a very tight lower bound leading branch-and-cut tree to converge quickly. Further, as shown in Table 5.8, the computational time can be significantly reduced by parallelization.

Table 4.6 Computational time comparison of DS-ASP and ASP

$F \rightarrow$	30				40				% Savings (vs. ASP)
γ	50		60		50		60		
k	DS-ASP	ASP	DS-ASP	ASP	DS-ASP	ASP	DS-ASP	ASP	
1	0.52 (0.19-1.06)	0.57 (0.49-0.73)	0.47 (0.44-0.66)	0.55 (0.27-0.61)	1.42 (0.55-2.21))	4 (2.23-5.79)	1 (0.75-1.28)	2.9 (2.34-4.21)	57.4
2	1.93 (0.78-3.93)	18.4 (2.56-60.93)	1.87 (0.93-2.50)	3.41 (0.79-4.84)	9.73 (2.80-14.22))	> 1312 (76.4 > 3600)	8.85 (4.25-19.43)	303 (23.31-1762)	> 98.6
3	9.69 (6.07-16.50)	189.63 (29.43-613.73)	7.93 (2.56-11.66)	12.38 (1.16-41)	166.87 (21.86-427.93))	> 3380 (1018-> 3600)	149.2 (19.29-419)	> 1567 (12.89-> 3600)	> 94

Table 4.7 Computational time comparison of DS-ASP and its parallel implementation

$F \rightarrow$	30		40	
k	DS-ASP	Parallel	DS-ASP	Parallel
1	0.4	0.12	0.96	0.18
2	1.6	0.22	7.1	1
3	10.3	0.88	111.7	14.6

4.4.3 DS-ASP in case of Unconstrained Position Shifting

Aircraft sequencing problem can be viewed as a job shop scheduling problem with release times and sequence-dependent processing times (but zero setup times) or sequence-dependent setup times (but zero processing times) (Beasley et al. (2000), Bianco et al. (2006)), where release times in job shop are equivalent to earliest times of arrival in ASP, and sequence-dependent processing/setup times to safety separation requirements. Thus, there is a promising avenue for the DS-ASP to be applied to various other important scheduling problems in transportation or other areas some of which are also listed in (Lahmar and Benjaafar 2007). However, CPS constraints may not always be imposed for these scheduling problems, and solving the DS-ASP to optimality in case of unconstrained position shifting (equivalent to very large value of k) is not computationally viable.

But, as shown next, an unconstrained sequence dependent scheduling problem can be solved using data-splitting algorithm with imposition of k , say $k = 10$, and achieve an objective function value close to optimality. There are two drivers behind this. First, makespan tends to converge as k increases. Second, empirically, DS-ASP is efficient in handling realistic instances of ASP for k up to 10. To illustrate this, the makespan decrease is recorded in Fig 4.1 and computation time increase in Fig 4.2 with increasing k (averaged for 10 instances of size 30 and 40). For instance size 30, average makespan of 2556 is obtained in just 74 seconds; whereas, directly solving it using solver takes 1156 seconds on average. Notably, 2556 is also the average optimal value. Similarly, for an instance size of 40, average makespan of 3397 is obtained in 544 seconds; whereas in the preset time limit of 3600 seconds, the best upper bound obtained using solver is 3431 (> 3397).

Fig. 4.1 Illustration of makespan decrease with increasing k Fig. 4.2 Illustration of solver time increase with increasing k

4.5 Summary

In this chapter, a data-splitting algorithm for ASP on single runway with the objective of maximizing throughput is proposed which provides real time optimal solutions for the large scale realistic instances; a first LP-based branch-and-cut approach to do so. Existing solution approaches, including dynamic programming, and direct application of commercial solvers are intractable under wide range of real-life scenarios. Various procedures are prescribed to further enhance the performance of the algorithm in form of fathoming the sibling instance pairs

and multiple splits. Our computational experiments demonstrate that an LP based branch-and-cut approach, applied on the above model, outperforms dynamic programming-based approaches in handling different problem scenarios, including that of mixed-traffic. The algorithm is easy to implement, and the run times are sufficiently small to enable real-time deployment. Finally, an extension is suggested to solve the various job shop scheduling problems which may not be restricted by CPS.

In the next chapter, ASP on dual runways system with the objective of maximizing throughput is investigated. Further, noticing the absence of tractable approaches in handling this problem which is more complex than single runway scheduling, DS-ASP is extended to solve this problem. Detailed algorithmic insights and various procedures to improve the computational times are presented.

Chapter 5

Aircraft Sequencing on Two Runways: A Data-Splitting Approach

5.1 Introduction

Motivated by the practical importance of ASP for multiple runways (m -ASP) and algorithmic limitations involved in its solution presented in Chapter 2, this chapter studies the 2-ASP (a special case of the m -ASP for two runways, operating independently) which is originally defined in Psaraftis (1980). The formulation for 2-ASP presented here accounts for both arrival only and mixed mode of operations, CPS constraint, wide time-windows, early landings/departures (an aircraft landing/departing before its target time) and the objective of maximizing total throughput from both runways. Time-window restrictions, mixed mode of operations and early landings are not considered in Psaraftis (1980) but make the problem more practically relevant. To the best of our knowledge, no efficient solution exists for solving the 2-ASP under the aforementioned scenarios.

As evident from the literature review presented in Chapter 2, dynamic programming and set partitioning based algorithms cannot be directly applied to the problem under consideration, because: (i) unlike in the 1-ASP case, minimizing makespan (equivalent to maximizing runway throughput) for the 2-ASP lacks the required optimal-substructure, thus prohibiting dynamic programming-based approaches; and (ii) when employing a branch-and-price approach, each sub-

problem belonging to a runway has to be solved independently, which is not viable under the CPS framework. As it happens, being a variant of the NP-hard problem, the solution methodology has to be either based on dynamic programming or *branch-and-bound*; the former does not seem promising because of the very structure of the problem.

Next, the ASP is formulated as a 0–1 mixed-integer program (MIP) which is an adaptation of Beasley et al. (2000)’s model, enhanced with the inclusion of the CPS and symmetry breaking constraints. Recognizing that varied scenarios of large-scale realistic instances of this problem cannot be solved optimally in real or near-real time by directly employing the MIP formulation, this chapter proposes an extension to data-splitting algorithm, called DS-ASP, which successfully solved the 1-ASP in Chapter 3 and 4, that optimizes flight sequences by a repeated application of the 0–1 MIP on smaller data sets, while ensuring the optimality of the entire set, and demonstrates the efficacy of this approach. Another important and intrinsic feature of the proposed data-splitting algorithmic framework is that it is pleasingly parallelizable, making it amenable to rapidly scalable computations.

There are many significant contributions to the literature that stem from this study:

1. The 0–1 MIP formulation presented here encapsulates the most realistic version of the 2-ASP, by simultaneously accounting for system safety, runway efficiency, and airline equity, as model constraints or via the objective function.
2. It provides the optimal solutions for large-scale realistic instances of the 2-ASP using a novel methodology at the complexity level unreachable by existing algorithms or commercial solvers.
3. To the best of our knowledge, it is the first LP-based branch-and-cut method to tractably solve this problem.
4. It provides a framework for the tractable solution of the m -ASP in case of more than two runways.
5. It promises an avenue that looks beyond the set partitioning based approaches for the problems rooted in VRPTW.

The remainder of this chapter is organized as follows. In Section 5.2, the 0–1 MIP formulation for the ASP is presented along with some preliminary computations that serve to highlight the inherent difficulty in solving the ASP for two independently operating runways via a straightforward MIP approach.

Next, the details of the data-splitting algorithm and its pseudo-code are described in Section 5.3. Detailed algorithmic concepts and analysis are provided that offer critical insights into the development of this method, and validate the computational performance of the algorithm. Then, Section 5.4 presents detailed computational results and analysis pertaining to large-scale (realistic) instances, and finally, Section 5.5 summarizes the contributions of this chapter.

5.2 Optimization Problem

2-ASP is formulated as a 0–1 mixed integer program, as detailed below.

Description of Index Sets and Parameters

$\mathcal{F}$: Set of all arriving and departing flights

$\mathcal{R}$: Set of all runways

E_i : Earliest time of arrival (departure) of aircraft i

T_i : Target time of arrival (departure) of aircraft i

L_i : Latest time of arrival (departure) of aircraft i

Δ_{ij} : Required safety separation (in seconds) at runway threshold, if flight i is ahead of flight j (see Table 1)

$seq(i)$: Position of flight i based on the FCFS sequence

k : Specified maximum position shifting (MPS) parameter

Decision Variables

$$x_{ij} = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in sequence} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ir} = \begin{cases} 1, & \text{if flight } i \text{ lands/depart on runway } r \\ 0, & \text{otherwise} \end{cases}$$

$$w_{ij} = \begin{cases} 1, & \text{if flight } i \text{ and flight } j \text{ land on the same runway} \\ 0, & \text{otherwise} \end{cases}$$

t_i = Scheduled time of arrival (departure) of flight i

z_r = Makespan of runway r

$$\mathbf{2-ASP :} \quad \text{Minimize} \quad z_1 \quad (5.1)$$

$$\text{subject to:} \quad x_{ij} + x_{ji} = 1, \quad \forall i < j, (i, j) \in \mathcal{F} \quad (5.2)$$

$$\sum_{r \in \mathcal{R}} y_{ir} = 1, \quad \forall i \in \mathcal{F} \quad (5.3)$$

$$w_{ij} \geq y_{ir} + y_{jr} - 1, \quad \forall i < j, (i, j) \in \mathcal{F}, \forall r \in \mathcal{R} \quad (5.4)$$

$$w_{ij} = w_{ji}, \quad \forall i < j, (i, j) \in \mathcal{F} \quad (5.5)$$

$$t_j \geq t_i + \Delta_{ij} w_{ij} - M_1(1 - x_{ij}), \quad \forall (i, j) \in \mathcal{F} \quad (5.6)$$

$$z_r \geq t_i - M_2(1 - y_{ir}), \quad \forall i \in \mathcal{F}, \forall r \in \mathcal{R} \quad (5.7)$$

$$z_1 \geq z_2 \quad (5.8)$$

$$-k \leq (n - \sum_{j \in \mathcal{F}, i \neq j} x_{ij}) - \text{seq}(i) \leq k, \quad \forall i \in \mathcal{F} \quad (5.9)$$

$$E_i \leq t_i \leq L_i, \quad \forall i \in \mathcal{F} \quad (5.10)$$

$$x_{ij}, w_{ij} \in \{0, 1\}, \forall (i, j) \in \mathcal{F}; y_{ir} \in \{0, 1\}, z_r, t_i \geq 0, \forall i \in \mathcal{F}, r \in \mathcal{R}. \quad (5.11)$$

In the above formulation, the objective function (5.1) seeks to minimize the makespan; constraint (5.2) enforces the order precedence relationship between flights i and j ; constraint (5.3) dictates that each plane is assigned to exactly one runway; constraints (5.4) ensures that the conditional relationships between the w and y variables are respected; constraint (5.5) is a symmetry constraint that maintains consistency of runway allocations; constraint (5.6) mandates the time-based separation requirements between flights at the runway, where $M_1 \equiv (L_i + \Delta s_{ij} - E_j)$, are satisfied; constraint (5.7) defines the makespan of runway r to be greater than or equal to the scheduled times of all flights allocated

to runway r , where $M_2 \equiv L_i$; constraint (5.8) breaks the symmetry by specifying that the last scheduled flight in the overall schedule must be assigned to the first runway; constraint (5.9) imposes the CPS constraint that an aircraft cannot be shifted by more than k positions from its initial (FCFS) position, where $n \equiv |\mathcal{F}|$; constraint (5.10) maintains the scheduled time of arrival (departure) at the runway to be between the earliest and latest times for each aircraft; and finally constraint (5.11) imposes binary and non-negativity restrictions on the (w, x, y) and (t, z) variables, respectively. Note that the ASP formulation described above can again be further improved by using variable-fixing strategies presented in Chapter 3.

To gain an understanding of the computational difficulty involved in solving Problem 2-ASP, the above optimization model (along with variable fixing strategies) is tested on several randomly generated large-scale (realistic) instances. The lack of responsiveness of this model (even when $k = 1$) can be easily observed from the computation times (in seconds) recorded in Table 5.1, which necessitates the development of an efficient solution technique for solving this problem. The preset time limit is 2 hours for instances of size 30 and 35, and 5 hours for instances of size 40.

Table 5.1 Computational times (sec) based on a direct implementation of Problem 2-ASP on large-scale instances

Instance	# of flights	Mode of Operation	MPS (k)	
			1	2
1	30	Arrival	5243	> 7200
2	35	Arrival	> 7200	> 7200
3	35	Arrival	> 7200	> 7200
4	40	Arrival	> 18000	> 18000
5	40	Arrival	> 18000	> 18000
6	40	Mixed	1544	> 18000

5.3 Implementation of DS-ASP

As discussed in previous chapters, there are two stages involved in the *data-splitting algorithm*. The first stage is *instance generation*, in which the original flight data-set is divided into several possible pairs of disjoint subsets of leading and the following aircraft. The second is the *solution stage*, in which each resulting subset pair is independently solved to determine its optimal solution. The obtained solutions are finally compared to get the overall optimum.

The basic framework or stages of DS-ASP and its drivers stay the same for all the possible applications; however, the procedures at both the stages may be

different for each application. This procedural difference is intrinsic for DS-ASP in 2-ASP as compared to 1-ASP proposed in Chapter 3. In the instance generation stage (the details are given below), the particular category of aircraft needs to be fixed at last position on both the runways, thereby resulting in a comparatively higher number of instance pairs as compared to 1-ASP where only one aircraft type is fixed. Further, in the solution stage, the search space of the following set is dictated by makespans of the leading set on both the runways, as compared to 1-ASP where only one makespan dictates the search space. Obviously, the resulting computational complexity is much higher in case of 2-ASP.

A more detailed description of the algorithmic stages is given below. In Section 5.3.1 the procedure is first described for a segregated mode of operation involving only arriving traffic, which is then extended to the mixed mode of operations in Section 5.3.4.

5.3.1 Algorithmic outline for arriving traffic

Stage A: Preparation of instance pairs

Let n denote the number of aircraft positioned at $\{1, \dots, m-1, m, m+1, \dots, n\}$ in FCFS order. After splitting this flight data into two halves, the leading set A and following set B are composed of aircraft at positions $\{1, \dots, m\}$ and $\{m+1, \dots, n\}$, respectively. Owing to the CPS constraint, as the only aircraft between $(m-k+1)$ and $(m+k)$ positions can crossover between the leading/following sets, there are only $\binom{2k}{k}$ such pairs. Note that both the *aircraft type* and the *arrival time* of the last scheduled flight on each runway in the leading set affect the makespan of the following set (or the overall makespan) because aircraft in the following set have to maintain the required safety separations from their counterparts in the leading set. In order to account for the effect of *aircraft type*, the number of pairs is further enlarged by fixing all possible aircraft in the leading set that can be positioned last on each runway. (The effect of the arrival time will be accounted for in the solution stage.) Specifically, observing that only aircraft between positions $(m-k)$ to m can occupy the last position in set A , each pair (A, B) results in additional combinations, denoted as $(\dot{A}, B)$. Furthermore, each pair $(\dot{A}, B)$ gives rise to some more combinations $(\ddot{A}, B)$ after fixing each aircraft type which can be scheduled last on the second runway. These $(\ddot{A}, B)$ pairs are then

used by the DS-ASP algorithm in the solution stage. This scheme to develop all the pairs is illustrated in the example below.

Consider an instance with eight aircraft $\{H1, H2, H3, L4, L5, H6, S7, S8\}$, where each aircraft is represented by its aircraft type (H : Heavy, L : Large, S : Small) and its relative position in the FCFS sequence. Assuming $k = 2$, there are $\binom{4}{2} = 6$ resulting pairs (A, B) , which are given by:

$$\begin{aligned} A &= \{H1, H2, H3, L4\}, B = \{L5, H6, S7, S8\} \\ A &= \{H1, H2, H3, L5\}, B = \{L4, H6, S7, S8\} \\ A &= \{H1, H2, H3, H6\}, B = \{L4, L5, S7, S8\} \\ A &= \{H1, H2, L4, L5\}, B = \{H3, H6, S7, S8\} \\ A &= \{H1, H2, L4, H6\}, B = \{H3, L5, S7, S8\} \\ A &= \{H1, H2, L5, H6\}, B = \{H3, L4, S7, S8\}. \end{aligned}$$

For each of these listed pairs (A, B) , permuting flights within set A , predicated on the aircraft occupying the last position on the first runway (labeled as l_r , where l and r denote the last flight and runway, respectively), results in several additional combinations, denoted as $(\dot{A}, B)$. As an example, for the case of $A = \{H1, H2, H3, L4\}$ and $B = \{L5, H6, S7, S8\}$, such permutations within set A result in the following three pairs:

$$\begin{aligned} \dot{A} &= \{H1, H2, H3, L4_1\}, B = \{L5, H6, S7, S8\}, \\ \dot{A} &= \{H1, H2, L4, H3_1\}, B = \{L5, H6, S7, S8\}, \\ \dot{A} &= \{H1, H3, L4, H2_1\}, B = \{L5, H6, S7, S8\}. \end{aligned}$$

Note that, for case $A = \{H1, H2, H3, H6\}$ and $B = \{L4, L5, S7, S8\}$, as aircraft $H6$ cannot be scheduled before position 4 in the sequence (as this would violate the CPS constraint), no further pairings are feasible. Therefore, in the above example, the total number of all possible instance pairs $(\dot{A}, B)$ is twelve. Furthermore, if pair (A, B) or $(\dot{A}, B)$ violates the ELW rule, then some of the pairs, e.g. $\dot{A} = \{H1, H3, L4, H2_1\}$ and $B = \{L5, H6, S7, S8\}$, can be discarded. As a result of this pruning exercise, for the solution phase of this algorithm, only five pairs of $(\dot{A}, B)$ instances are left as compared to the originally derived twelve instances (see Chapter 3 for more details). Next, for each of these pairs $(\dot{A}, B)$, several additional derivative pairs $(\ddot{A}, B)$ are generated based on the aircraft type which can be scheduled last on the second runway. As an example, for the pair: $\dot{A} = \{H1, H2, L4, H3_1\}$, $B = \{L5, H6, S7, S8\}$, the resulting derivative $(\ddot{A}, B)$ pairs are listed as follows:

$$\begin{aligned}\ddot{A} &= \{H1, H2, L4_2, H3_1\}, B = \{L5, H6, S7, S8\}, \\ \ddot{A} &= \{H1_2, H2_2, L4, H3_1\}, B = \{L5, H6, S7, S8\}.\end{aligned}$$

It turns out that the landing order of the last scheduled aircraft on the second runway cannot be fixed in advance (see Fig 5.1 for illustration), hence multiple aircraft in $\ddot{A}$ of the same aircraft type can be the potential candidates (as in second pair $(\ddot{A}, B)$ above) for being scheduled last on the second runway. Additional set of constraints (5.12)-(5.14) are added to $Opt_{\ddot{A}}$ to ensure that aircraft landing at last position should be the one from set $\ddot{\mathcal{C}}$ that consists of the candidate aircraft of a certain type; ξ_j is the binary decision variable which takes value zero when aircraft j is scheduled to last.

Fig. 5.1 The landing order (right to left) of the last aircraft on second runway is second in the left figure whereas it is third in the right figure

$$x_{ij} \geq y_{i2} + y_{j2} - 1 - \xi_j \quad \forall i \in \ddot{A}, j \in \ddot{\mathcal{C}}, i \neq j \quad (5.12)$$

$$\sum_{j \in \ddot{\mathcal{C}}} \xi_j = |\ddot{\mathcal{C}}| - 1 \quad (5.13)$$

$$\xi_j \geq y_{j1} \quad \forall j \in \ddot{\mathcal{C}} \quad (5.14)$$

Constraint (5.12), active only when $\xi_j = 0$, ensures that any aircraft i landing on the second runway must do so before j ; constraint (5.13) ensures that only one aircraft is scheduled to last on the second runway; and finally, constraint (5.14) ensures that ξ_j variables for aircraft landing on the first runway cannot take zero value. Note that by taking advantage of the ELW rule, the size of set $\ddot{\mathcal{C}}$ can be restricted to further strengthen the above formulation.

Stage B: Solving the instance pairs

Let $(\ddot{z}_1, \ddot{z}_2)$ denote an *optimal state* for leading set $\ddot{A}$, where $\ddot{z}_r$ denotes the optimal makespan on runway r with $\ddot{z}_1 \geq \ddot{z}_2$. When optimizing the leading set of flights, it is always desirable to keep the makespan on the two runways as small as possible so that a feasible solution for the following set of flights can be

found leading to minimum makespan. However, there is an inverse relationship between $\check{z}_1$ and $\check{z}_2$, i.e., decreasing one may increase the other, and as a result, a brute-force method that evaluates all possible optimal states $(\check{z}_1, \check{z}_2)$ remains the only avenue to find the optimal solution.

However, the number of such states can be very large and evaluating all possibilities is a computationally intensive task. As flights in the following set are constrained to land at or after $\check{z}_1$, the key to deciding optimality of the following set is not only to reduce the $\check{z}_1$ -value but also increase the gap $(\check{z}_1 - \check{z}_2)$, as the aircraft in the following set that land on the second runway can take advantage of this gap to land as early as possible. (For example, in Example 5.1 below, even though optimal schedule in Figure 5.5 has higher $\check{z}_1 (= 230)$ than in Figure 5.3 or 5.4, its higher gap $(= 130)$ than its other two counterparts has helped it in achieving optima.) This observation lays the foundation of the algorithmic design, wherein a nonnegative continuous indicator variable δ which reflects the to-be-obtained advantage due to the $(\check{z}_1 - \check{z}_2)$ gap, is embedded in the optimization of the following set. This δ -variable may take on a non-zero value, which indicates that adding δ to the existing gap $\check{z}_1 - \check{z}_2$, i.e., optimizing with respect to the state $(\check{z}_1, \check{z}_2 - \delta)$ may result in a better optimal solution, whereas a zero value of δ acts as the stopping criteria, indicating that no further states resulting from the pair $(\check{A}, B)$ need to be evaluated. Furthermore, whenever δ is found to be non-zero, the leading set is re-optimized to seek a new improved state $(\check{z}_1, \check{z}_2)$.

Next, the optimization problems for set $\check{A}$ and B are defined. The discussion below will go over all the important considerations while framing the objective functions and constraints for each optimization problem.

1. Optimization problem for set $\check{A}$: $\check{z}_1$ and $\check{z}_2$ are the points on Pareto frontier as there is an inverse relationship between $\check{z}_1$ and $\check{z}_2$ (one cannot be made better off without making the other worse). On this frontier, start from the point with minimum $\check{z}_1$ and proceed to the immediate next point as movement is desired in the direction where the gap $\check{z}_1 - \check{z}_2$ increases. For example, Example 5.1 explores through three Pareto-Optimal Points. This process is continued until no further Pareto-Optimal point can be obtained or stopping criteria is met. Next, the objective function is formulated as minimizing $z_1 + \theta z_2$, with θ is sufficiently small so as to ensure that the starting point is one with minimum $\check{z}_1$, as well as, $\check{z}_2$ should not be optimized at the expense of $\check{z}_1$; at each subsequent iteration, an

extra constraint (5.15) is added to move to the next point. Figure 5.2 represents this frontier, where Point 1 is the starting state and Point 2 is the next state. Note that Point $\hat{1}$ (not a pareto optimal point) is not attained during the course of the optimization.

$$z_2 \leq \ddot{z}_2 - \alpha, \quad \alpha = 1, \text{ assuming all input parameters are integers} \quad (5.15)$$

Fig. 5.2 Pareto frontier consisting of $\ddot{z}_1$ and $\ddot{z}_2$

Finally, the optimization problem for set $\ddot{A}$ can be written as:

$$Opt_{\ddot{A}} \equiv \min z_1 + \theta z_2, \text{ s.t. (5.2) -- (5.11), (5.12) -- (5.15).}$$

2. Optimization problem for set B . Similar to 2-ASP, the objective here is also to minimize the overall makespan. However, unlike 2-ASP, only a single makespan variable z is used as symmetry is lost between the two runways for the following set. For example, in Example 5.1 (Figure 5.5), flipping the runway for each aircraft in the optimal schedule of the leading set doesn't affect the final solution (optimal schedule of following set adjusts accordingly as it is solved after the leading set), whereas doing it for the following set does. (The optimal value of z is also the optimal makespan of pair $(\ddot{A}, B)$.) In addition, constraints (5.16)–(5.19) are added to ensure that flights in set B follow set $\ddot{A}$, where: (5.16) is a simple makespan constraint; (5.17) ensures that the scheduled time of any flight i in the following set should be greater than or equal to $\ddot{z}_1$; (5.18) (resp. (5.19)) ensures that the scheduled time of any flight i on first runway (resp. second runway) in the following set should be safely separated from the last

aircraft l_1 (resp. l_2) in the leading set.

$$z \geq t_i \quad \forall i \in B \quad (5.16)$$

$$t_i \geq \ddot{z}_1 \quad \forall i \in B \quad (5.17)$$

$$t_i \geq \ddot{z}_1 + y_{i1} \Delta_{l_1 i} \quad \forall i \in B \quad (5.18)$$

$$t_i \geq \ddot{z}_2 + y_{i2} \Delta_{l_2 i} \quad \forall i \in B \quad (5.19)$$

This particular problem, referred to as Opt_B can be succinctly stated as:

$$Opt_B \equiv \min z, \text{ s.t. } (5.2) - (5.6), (5.9) - (5.11), (5.16) - (5.19).$$

Furthermore, the optimization problem on set B to calculate δ is formulated by modifying constraint (5.19) as follows. The embedded δ ensures that the effective makespan of leading set on the second runway can be lowered to any extent.

$$t_i \geq \ddot{z}_2 + y_{i2} \Delta_{l_2 i} - \delta \quad \forall i \in B \quad (5.20)$$

This particular problem, referred to $\delta - Opt_B$ can be stated as:

$$\delta - Opt_B \equiv \min z + \varepsilon_1 \delta, \text{ s.t. } (5.2) - (5.6), (5.9) - (5.11), (5.16) - (5.18), (5.20),$$

where $\varepsilon_1 < \frac{1}{\max \Delta}$ (Proposition 5.1). (Note that $\max \Delta$ is the maximum of all separations in Table 1.1.)

Proposition 5.1. ε_1 less than $\frac{1}{\max \Delta}$ ensures that overall makespan can be reduced by minimum possible amount.

PROOF: Assuming all parameters are integer, the minimum change in makespan is integral and hence equal to 1. Therefore, for any value of δ , ε_1 should maintain the relation $\max(\varepsilon_1 \delta) < 1$, which further implies $\varepsilon_1 \max \Delta < 1$ or $\varepsilon_1 < \frac{1}{\max \Delta}$.

The pseudo-code of the proposed data-splitting algorithm is listed below.

Algorithm 5.1: Pseudo-code for the proposed data-splitting algorithm

- 1: Set incumbent $\leftarrow +\infty$, $\Pi \leftarrow \emptyset$ (Π is the incumbent decisions), and $S = \{\text{set of all pairs } (\ddot{A}, B)\}$.

-
- 2: If $S = \emptyset$, **STOP**; return incumbent and Π as the best solution. Else, go to Step 3.
 - 3: Arbitrarily select one instance pair $(\check{A}, B)$ from S . Set $\check{z}_2 \leftarrow +\infty$.
 - 4: Solve $Opt_{\check{A}}$. Let $(\check{z}_1, \check{z}_2)$ be the optimal solution state and denote (l_1, l_2) as the last scheduled aircraft on the first and second runways.
 - 5: Solve $\delta - Opt_B$. Let $\bar{z}_\delta$ be the value of the optimal solution. If $\delta > 0$ go to Step 6; else set $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 7.
 - 6: Solve Opt_B . Let $\bar{z}$ be the value of the optimal solution. If $\bar{z} < \text{incumbent}$, set $\text{incumbent} \leftarrow \bar{z}$. Go to Step 4.
 - 7: If $\bar{z} < \text{incumbent}$, set $\text{incumbent} \leftarrow \bar{z}$ and update Π with the optimal solution of pair $(\check{A}, B)$. Update $S \leftarrow S \setminus \{\check{A}, B\}$. Go to Step 2.
-

EXAMPLE 5.1 A simple example is now presented to illustrate the algorithm. Consider an instance pair $(\check{A}, B)$ with $(l_1, l_2) = \{H1, H2\}$ and $B = \{H3, S4\}$, where $H1, H2, H3$ are heavy arrivals and $S4$ is small arrival. Further, earliest time of arrival is set to be zero for each flight and latest time is set to be 1800 seconds. Now, consider three iterations as the algorithm moves towards convergence wherein optimal states $(\check{z}_1, \check{z}_2)$ (the solution of $Opt_{\check{A}}$) at the end of each iteration are $(200, 180)$, $(220, 140)$, and $(230, 100)$, respectively. From the results of Iteration 1 shown in Fig 5.3, non-zero value of δ suggests that an improvement in makespan can be obtained after lowering the makespan of leading set on second runway. The makespan is 376 in Fig 5.3a (optimization without δ) when $\check{z}_2 = 180$; it is 296 in case of Fig 5.3b where makespan of leading set on second runway is lowered to $\check{z}_2 - \delta = 100$. Fig 5.4 then presents Iteration 2 where $\delta = 20$. Finally, from Iteration 3 (Fig 5.5), stopping criteria $\delta = 0$ is met, and optima is concluded after comparing all the $\bar{z}$. The final optima is 326.

REMARK 5.1 From the above example, it can be observed that the optimal solution of the pair corresponds to $\delta = 0$. However, this may not always be the case, e.g., if the optimal state in the third iteration is $(310, 100)$ and not $(230, 100)$. Clearly, $\bar{z}$ corresponding to the state that leads to $\delta = 0$ is 406, whereas the new minimum is 336. $\square$

Fig. 5.3 Iteration 1: $(\bar{z}_1, \bar{z}_2) = (200, 180)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$ Fig. 5.4 Iteration 2: $(\bar{z}_1, \bar{z}_2) = (220, 140)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$ Fig. 5.5 Iteration 3: $\delta - Opt_B$ run with $(\bar{z}_1, \bar{z}_2) = (230, 100)$, $\Delta_{HS} = 196$, $\Delta_{HH} = 96$

5.3.2 Algorithmic Enhancements and Improvements

This section presents the enhancements to increase the computational efficiency of the proposed DS-ASP algorithm through fathoming based on the bound and combinatorial analysis. Let $(\bar{z}_1, \bar{z}_2)$ denotes the current optimal state.

A. Fathoming based on bound

As is evident from the foregoing discussion, at any iteration corresponding to a particular pair $(\bar{A}, \bar{B})$, the following set is optimized sequentially in two ways: (i) embedding δ , which leads to the optimal solution $\bar{z}_\delta$; and (ii) without embedding δ , which provides optimal solution $\bar{z}$. Embedding δ effectively relaxes the

constraint imposed due to makespan $\bar{z}_2$, and $\bar{z}_1$ determines the overall search space for the following set. Therefore, it can be concluded that: (i) $\bar{z}_\delta$ is the lower bound on $\bar{z}$; (ii) with each iteration, $\bar{z}_\delta$ increases since $\bar{z}_1$ increases; and (iii) (from the first two conclusions) subsequent iterations for a particular pair are fathomed if $\bar{z}_\delta$ is greater than or equal to the minimum of all the calculated $\bar{z}$ so far.

B. Calculating δ and $\bar{z}_\delta$ using values from previous optimization run

Since $\bar{z}_\delta$ depends only on $\bar{z}_1$ and not on the last aircraft and makespan of the second runway, current optimal decision (allocation and sequencing) can be used in finding $\bar{z}_\delta$ and δ values for subsequent iterations of the current pair, as well as of any other pair for which last aircraft on first runway is same as the current pair; however, the current optimal decision of set B must not *get better* (equivalently, length of the sequence reduces), as well as, *remain feasible* even as $\bar{z}_1$ varies (from its original value). To check whether it gets better, $\delta - Opt_B$ is modified with two more indicator parameters μ_1 and μ_2 which take on the value zero if optimal decision of set B does not get better while varying $\bar{z}_1$. Fortunately, for the problem under consideration, lower inter-arrival times (or closely spaced target times) of aircraft and wider time windows—which are associated with higher problem complexity—lead to $\mu_1 = 0$ and $\mu_2 = 0$. (Its rationale and the proof are given next.) Finally, feasibility is automatically ensured when the current decision is used to solve the $\delta - Opt_B$ for any subsequent iteration or pair. The above-mentioned modification can be stated as follows:

$$\text{Minimize} \quad \sum_{i \in B} z - \mu_1 + \mu_2 + \varepsilon_1 \delta + \varepsilon_2 (\mu_1 + \mu_2) \quad (5.21)$$

$$t_i \geq \bar{z}_1 + y_{i1} \Delta_{l_1 i} + \mu_1 - \mu_2 \quad \forall i \in B \quad (5.22)$$

$$t_i \geq \bar{z}_2 + y_{i2} \Delta_{l_2 i} - \delta + \mu_1 - \mu_2 \quad \forall i \in B \quad (5.23)$$

$$t_i \geq \bar{z}_1 + \mu_1 - \mu_2 \quad \forall i \in B \quad (5.24)$$

Note that constraints (5.22) and (5.23) are obtained after the respective modifications of constraints (5.18) and ((5.20)). The modified $\delta - Opt_B$ can be stated as:

$\delta - Opt_B^m \equiv \min z - \mu_1 + \mu_2 + \varepsilon_1 \delta + \varepsilon_2(\mu_1 + \mu_2)$, s.t. (5.2) – (5.6), (5.9) – (5.11), (5.16), (5.22) – (5.24).

Next, the rationale and mathematical proofs of the above modification are provided.

1. Rationale. As $\tilde{z}_1$ is varied from its original value, sequence of the following set may get pulled in or pushed out and a new and better sequence may emerge. When the following sequence is pushed out (as a result of increase in $\tilde{z}_1$ value), a better following sequence will only emerge if that sequence is currently infeasible due to violation of earliest landing time (E). However, that is unlikely in case of lower inter-arrival times (as compared to average separation of the following sequence) since it may result in late landings of the flights in following sequence.

From Fig 5.6, it can be seen that a better sequence P' which earlier was infeasible (due to violation of $E_{2'}$) becomes feasible and optimal when the sequence is pushed out. However, as mentioned above, this is unlikely to happen in case of closely spaced earliest (or target) times in peak traffic scenarios.

Fig. 5.6 Sequence change in case of widely spaced earliest times (the time-window of each flight is indicated with oval bracket)

Similarly, when the following sequence is pulled in (with a reduction in $\tilde{z}_1$ value), a better sequence can only occur if that sequence was infeasible (initially) due to a violation of the latest landing time (L). However, since time-windows are wider, that is unlikely to occur also. In summary, in case of lower inter-arrival times coupled with wide time-windows, a better sequence of the following set is unlikely to emerge as a value of $\tilde{z}_1$ is varied. It is illustrated next.

From Fig 5.7, it can be seen that a better sequence P which earlier was infeasible (due to violation of L_2) becomes feasible and optimal when the sequence is pulled in. However, as mentioned above, this is unlikely to happen in case of wide time-windows.

Fig. 5.7 Sequence change in case of narrow time-windows, the time-window of each flight is indicated with oval bracket

2. Proof. Here the proof is provided that μ_1 and μ_2 take non-zero value when better optimal decision results with change in makespan from $\bar{z}_1$. Let $\hat{z}_\delta$ denotes the optimal makespan when $\bar{z}_1$ varies.

Proposition 5.2. $\mu_1 > 0$ in the problem $\delta - Opt_B^m$ implies $\hat{z}_\delta < \bar{z}_\delta + \mu_1$. Similarly, $\mu_2 > 0$ implies $\hat{z}_\delta < \bar{z}_\delta - \mu_2$.

PROOF: Now, optima of $\delta - Opt_B^m$ occurring at $\bar{z}_1 + \mu_1$ implies,

$$\begin{aligned} \bar{z}_\delta + \varepsilon_1 \delta &\geq \hat{z}_\delta - \mu_1 + \varepsilon_2 \mu_1 + \varepsilon_1 \hat{\delta} \\ \Rightarrow \hat{z}_\delta &\leq \bar{z}_\delta + \mu_1 - \varepsilon_2 \mu_1 + \varepsilon_1 (\delta - \hat{\delta}) \\ \Rightarrow \hat{z}_\delta &\leq \bar{z}_\delta + \mu_1 (1 - (\varepsilon_2 - \frac{\varepsilon_1 (\delta - \hat{\delta})}{\mu_1})) \\ &\Rightarrow \hat{z}_\delta < \bar{z}_\delta + \mu_1 \end{aligned}$$

where constants ε_1 and ε_2 are preset tolerances such that $\varepsilon_2 - \frac{\varepsilon_1 (\delta - \hat{\delta})}{\mu_1}$ always remains a reasonably small positive quantity; being positive is important to maintain the inequality $\hat{z}_\delta < \bar{z}_\delta + \mu_1$, while being reasonably small is important to ensure μ_1 takes on a non-zero value for any occurrence of better decision. Proof of the second part can be given in a similar manner.

From the above proposition, the following corollary results,

Corollary 5.1. $\mu_1 = 0$ and $\mu_2 = 0$ in the optimal solution of $\delta - Opt_B^m$ implies that the optimal decision does not get better as $\bar{z}_1$ varies, i.e., $\hat{z}_\delta \geq \bar{z}_\delta \pm \theta$. The inequality exists as varying $\bar{z}_1$ to a greater degree may render the current decision infeasible.

Proposition 5.3. Constants ε_1 and ε_2 are related as $\varepsilon_2 + \varepsilon_1 \max \Delta = c$ and $\varepsilon_2 - \varepsilon_1 \max \Delta \geq 0$, where $c > 0$ is a user defined tolerance.

PROOF: Clearly, from the foregoing discussion, $(\varepsilon_1, \varepsilon_2)$ should be such that $\max(\varepsilon_2 - \frac{\varepsilon_1(\delta - \hat{\delta})}{\mu_1})$ is equal to c , and $\min(\varepsilon_2 - \frac{\varepsilon_1(\delta - \hat{\delta})}{\mu_1})$ is greater than or equal to zero. Now,

$\max(\varepsilon_2 - \frac{\varepsilon_1(\delta - \hat{\delta})}{\mu_1}) = c \Rightarrow \varepsilon_2 + \varepsilon_1 \frac{\max(|\delta - \hat{\delta}|)}{\min \mu_1} = c \Rightarrow \varepsilon_2 + \varepsilon_1 \max \Delta = c$,
as $\max(|\delta - \hat{\delta}|) = \max \Delta$, and $\min \mu_1 = 1$ (minimum change in makespan of leading set is 1). Similarly,

$$\min(\varepsilon_2 - \frac{\varepsilon_1(\delta - \hat{\delta})}{\mu_1}) \geq 0 \Rightarrow \varepsilon_2 - \varepsilon_1 \frac{\max(|\delta - \hat{\delta}|)}{\min \mu_1} \geq 0 \Rightarrow \varepsilon_2 - \varepsilon_1 \max \Delta \geq 0.$$

Note that $\varepsilon_1 < \frac{1}{\max \Delta}$ (from Proposition 5.1).

The pseudo-code of the enhanced data-splitting algorithm is listed below.

Algorithm 5.2: Pseudo-code for the enhanced data-splitting algorithm

- 1: Set incumbent $\leftarrow +\infty$, $\Pi \leftarrow \emptyset$, and $\dot{S} = \{\text{set of all pairs } (\dot{A}, B)\}$.
 - 2: If $\dot{S} = \emptyset$, **STOP**; return incumbent and Π as the best solution; else, go to Step 3.
 - 3: Arbitrarily select one instance pair $(\dot{A}, B)$ from $\dot{S}$, set $q \leftarrow 0$. Let $\ddot{S} = \{\text{set of all pairs } (\ddot{A}, B)\}$ corresponding to $(\dot{A}, B)$.
 - 4: If $\ddot{S} = \emptyset$, update $\dot{S} \leftarrow \dot{S} \setminus \{\dot{A}, B\}$, go to Step 2. Else, arbitrarily select one instance pair $(\ddot{A}, B)$ from $\ddot{S}$, set $\ddot{z}_2 \leftarrow \infty$.
 - 5: Solve $Opt_{\ddot{A}}$. Let $(\ddot{z}_1, \ddot{z}_2)$ be the optimal solution state and denote (l_1, l_2) as the last scheduled aircraft on the first and second runways. If $q = 0$, set $q \leftarrow 1$, go to Step 6; else, go to Step 7.
 - 6: Solve $\delta - Opt_B^m$. Let $\bar{\Pi}$ be the set of x - and y - optimal decisions, and let $\bar{z}_\delta$ is the value of optimal solution. If $\bar{z}_\delta - \max(0, \mu_1) + \max(0, \mu_2) \geq$ incumbent, update $\ddot{S} \leftarrow \ddot{S} \setminus \{\ddot{A}, B\}$, go to Step 4. Else, if $\mu_1 = 0$ and $\mu_2 = 0$, go to Step 8; else go to Step 7.
 - 7: If $\mu_1 = 0$ and $\mu_2 = 0$, solve $\delta - Opt_B$ using $\bar{\Pi}$; else, solve without using $\bar{\Pi}$. Let $\bar{z}_\delta$ be the value of optimal solution. If $\bar{z}_\delta \geq$ incumbent, update $\ddot{S} \leftarrow \ddot{S} \setminus \{\ddot{A}, B\}$, go to Step 4; else go to Step 8.
 - 8: If $\delta > 0$ go to Step 9. Else set $\bar{z} \leftarrow \bar{z}_\delta$; go to Step 10.
 - 9: Solve Opt_B . Let $\bar{z}$ be the value of optimal solution. If $\bar{z} < \text{incumbent}$, set incumbent $\leftarrow \bar{z}$. Go to Step 5.
 - 10: If $\bar{z} < \text{incumbent}$, set incumbent $\leftarrow \bar{z}$ and update Π with the optimal solution of pair $(\ddot{A}, B)$, update $\ddot{S} \leftarrow \ddot{S} \setminus \{\ddot{A}, B\}$, go to Step 4.
-

5.3.3 Multiple Data-Split

In the computational experiments with single split, it is observed that the DS algorithm, even when parallelized, is not able to provide an efficient optimal solution for larger-sized computationally complex instances, e.g. when $n = 60$, as single split results in either set still being of a quite large size. In an attempt to address this issue, splitting of the original set at multiple points is investigated where communication now needs to be established amongst several splits originating from the same pair $(\ddot{A}, B)$. As part of the multiple split framework, the entire set is recursively split into a leading set of specified size, say ϕ_1 , and a following set, until a following set of specified size, say ϕ_2 , is obtained. The entire process can be mathematically expressed as:

$$\ddot{A}, B \longrightarrow \bigcup_{i \in \Pi(B)} \ddot{A}_i, B_i \longrightarrow \bigcup_{j \in \Pi(B_i)} \ddot{A}_j, B_j \longrightarrow \bigcup_{k \in \Pi(B_j)} \ddot{A}_k, B_k, \dots,$$

where $\Pi(S)$ represents all possible splits of the following set S , with CPS parameter k .

EXAMPLE 5.2 An instance of size 60 can be first split into pairs of leading set of size 15 and a following set of size 45, further, the following set from each pair can be split into various pairs of leading set of size 15 and following set of size 30; and finally, the following set from each of the last split pair can be further split into various pairs of leading set and following set of size 15.

As mentioned above, optimization of the leading set enumerates all the Pareto optimal points. In case of an initial leading set, it is sufficient to enumerate the points with $(z_1 \geq z_2)$ as scheduling space is symmetric. However, enumeration of Pareto optimal points with $(z_1 \leq z_2)$ is also required in case of the intermediate leading set as it is scheduled on an asymmetric space. Therefore, it is optimized twice: Opt_A^1 considers the points with $z_1 \geq z_2$, whereas Opt_A^2 considers the points with $z_1 \leq z_2$.

Since the intermediate leading set is also a following set of the previous split, it has to communicate the stopping criteria (δ) with the preceding set. However, since all the preceding sets are scheduled on an asymmetric space (except for the very first leading set), δ is calculated separately for both the runways. Constraints (5.25)–(5.27) (similar to constraints (5.17)–(5.19)) are added to its optimization problems, which are closed-form expressions in terms of binary parameters (η_1, η_2) , where δ on first runway (resp. second runway) is calculated corresponding to $\eta_2 = 1$ (resp. $\eta_1 = 1$). Further, when the intermediate

set is optimized without δ , η_1 and η_2 both set to zero; the extra term $|\eta_1 + \eta_2 - 1| \max(\hat{z}_1, \hat{z}_2)$ ensures that the flights in the intermediate leading set land after the maximum of makespan of the preceding set on both runways. Finally, $\hat{z}_r, \hat{l}_r$ denote the makespan values and last aircraft from the preceding set.

$$t_i \geq \eta_1 \hat{z}_1 + \eta_2 \hat{z}_2 + |\eta_1 + \eta_2 - 1| \max(\hat{z}_1, \hat{z}_2) \quad \forall i \in \ddot{A} \quad (5.25)$$

$$t_i \geq \hat{z}_1 + y_{i1} \Delta_{\hat{l}_1 i} - \eta_2 \delta \quad \forall i \in \ddot{A} \quad (5.26)$$

$$t_i \geq \hat{z}_2 + y_{i2} \Delta_{\hat{l}_2 i} - \eta_1 \delta \quad \forall i \in \ddot{A} \quad (5.27)$$

The two problems can now be summarized as:

1. $Opt_{\ddot{A}_1} \equiv \min z_1 + \theta_1 z_2 + (\eta_1 + \eta_2) \theta_2 \delta$, s.t. (5.2) – (5.11), (5.12) – (5.14), $z_2 \leq \ddot{z}_2 - 1$, (5.25) – (5.27), where θ_1 is sufficiently small weight and $\theta_2 < \frac{\theta_1}{\max \Delta}$.
2. $Opt_{\ddot{A}_2} \equiv \min z_2 + \theta_1 z_1 + (\eta_1 + \eta_2) \theta_2 \delta$, s.t. (5.2) – (5.7), $z_2 \geq z_1$, (5.9) – (5.11), (5.12) – (5.14), $z_1 \leq \ddot{z}_1 - 1$, (5.25) – (5.27), where θ_1 is sufficiently small weight and $\theta_2 < \frac{\theta_1}{\max \Delta}$.

Note that constraints (5.12) to (5.14) are modified for second runway in $Opt_{\ddot{A}_2}$.

Similar to the intermediate leading set, two different optimizations problems are defined for the final following set as its preceding set is scheduled on an asymmetric space:

1. $\delta - Opt_B^1$: Equivalent to $\delta - Opt_B$.
2. $\delta - Opt_B^2$: The δ -constraint in this problem should now evaluate the desired gap $\ddot{z}_2 - \ddot{z}_1$, hence the constraint (5.17) and constraint (5.20) are modified into (5.28) and (5.29) respectively. The rest of the problem is the same as $\delta - Opt_B$.

$$t_i \geq \ddot{z}_2 \quad \forall i \in B \quad (5.28)$$

$$t_i \geq \ddot{z}_1 + y_{i1} \Delta_{l_1 i} - \delta \quad \forall i \in B \quad (5.29)$$

The pseudo-code is presented in Appendix C. (Note that the algorithmic enhancements are ignored in the code for the sake of simplicity.)

5.3.4 Extending the Algorithm to Mixed Traffic

When FCFS sequence consists of both arrival and departure aircraft, and every aircraft has a CPS constraint with respect to its FCFS sequence position, the data-splitting algorithm can be extended to the case of mixed traffic. When dealing with the segregated case of only arriving traffic, as the triangular inequality with respect to the safety separation distance matrix is always satisfied, it is sufficient to maintain the separation of the first aircraft in set B from the last aircraft in $\ddot{A}$; however, the triangular inequality may be violated in case of mixed-traffic for the following instances:

Case 1: On a given runway, of the last two flights i and j in set $\ddot{A}$, the second last is a heavy arrival, followed by a departure of any aircraft type, which is then followed by a large arrival k (from set B) on the same runway.

Case 2: On a given runway, of the last two flights in set $\ddot{A}$, the second last aircraft is a heavy arrival, followed by a departure of any type, which is then followed by a small arrival k (from set B) on the same runway.

In Case 1 (based on the given time-separation matrix (Table 1.1)), $\Delta_{ij} + \Delta_{jk} = 135$ seconds, whereas $\Delta_{ik} = 157$, resulting in a violation of 22 seconds. Similarly, violation of 61 seconds occurs in the latter case. Thus, in order to ensure that such violations do not occur, the original pair $(\ddot{A}, B)$ consisting of a departure at last position in $\ddot{A}$ on either runway is replaced with two new pairs, along with the necessary *non-triangular* separation constraints. The first pair fixes the heavy aircraft at second last position in $\ddot{A}$, and second pair does not allow any heavy aircraft at this position; there is no change for flights in set B . To illustrate this, consider the example data given in Section 3.1 ($\{H1, H2, H3, L4, L5, H6, S7, S8\}$), where $H2$ (heavy arrival) is replaced with $\bar{H}2$ ($\bar{\bullet}$ denotes departure), $H3$ with $L3$ (large arrival), $L4$ with $\bar{L}4$ (large departure); the remaining flights remain the same as before. In case of $\ddot{A} = \{H1, \bar{H}2, L3_2, \bar{L}4_1\}$ and $B = \{L5, H6, S7, S8\}$, the following additional derivative pairs would need to be generated as aircraft $\bar{L}4$ is scheduled at the last position (on first runway):

$$\begin{aligned} \textbf{Pair 1: } \ddot{A} &= \underbrace{\{[-, -, H1, \bar{L}4_1]\}}_{\text{Runway 1}}, \underbrace{\{[-, -, L3_2]\}}_{\text{Runway 2}}, B = \{L5, H6, S7, S8\} \\ \textbf{Pair 2: } \ddot{A} &= \underbrace{\{[-, -, \bar{H}2, \bar{L}4_1]\}}_{\text{Runway 1}}, \underbrace{\{[-, -, L3_2]\}}_{\text{Runway 2}}, B = \{L5, H6, S7, S8\} \end{aligned}$$

Furthermore, similar instances result if the aircraft scheduled last on the second runway is a small or large departure. Next, similar to the selection of last aircraft on second runway from a certain category, the penultimate aircraft on either runway cannot be fixed manually as the position of the aircraft is not known in advance. Therefore, the following extra set of constraints are defined to facilitate this.

To ensure that the second last aircraft on the first runway belongs to the set $\mathcal{C}$ (consisting of the eligible aircraft from a certain category), constraints (5.30)-(5.32) (similar to (5.12)-(5.14) with some notational changes) are included in $Opt_{\bar{A}}$. Further, let ψ_j be the binary indicator variable which takes value zero when aircraft j is scheduled last, and l_1 be the last aircraft on the first runway.

$$x_{ij} \geq y_{i1} + y_{j1} - 1 - \psi_j \quad \forall i \in \bar{A}, \quad j \in \mathcal{C}, \quad i \neq j, \quad i \neq l_1 \quad (5.30)$$

$$\sum_{j \in \mathcal{C}} \psi_j = |\mathcal{C}| - 1 \quad (5.31)$$

$$\psi_j \geq y_{j2} \quad \forall j \in \mathcal{C} \quad (5.32)$$

To ensure that the second last aircraft on the second runway belongs to a certain category is even trickier as the last scheduled aircraft on second runway is also unknown. Constraints (5.33)-(5.36) are included in $Opt_{\bar{A}}$ to select last aircraft from the eligible set $\mathcal{C}$. Further, let $\bar{\mathcal{C}}$ represent the set of aircraft eligible to land at last position on the second runway, and ζ_j be the binary indicator variable which takes the value zero when flight j is scheduled second last on the second runway.

$$x_{ij} \geq y_{i2} + y_{j2} - 1 - \zeta_j \quad \forall i \in \bar{A}, \quad j \in \bar{\mathcal{C}}, \quad i \notin \mathcal{C}, \quad i \neq j \quad (5.33)$$

$$\sum_{j \in \bar{\mathcal{C}}} \zeta_j = |\bar{\mathcal{C}}| - 1 \quad (5.34)$$

$$\zeta_j \geq y_{j1} \quad \forall j \in \bar{\mathcal{C}} \quad (5.35)$$

$$x_{hj} + \zeta_j \geq \xi_h - y_{h1} \quad \forall j \in \bar{\mathcal{C}}, \quad \forall h \in \mathcal{C}, \quad h \neq j \quad (5.36)$$

Constraints (5.33)-(5.35) are similar to constraints (5.30)-(5.32) with some notational changes described above. Further, constraint (5.36), which becomes active when $\xi_h = 1$ and $y_{h1} = 0$, is included to ensure that if $\zeta_j = 0$, j must be behind all the aircraft h belonging to $\mathcal{C}'$ which land on the second runway (implied by $y_{h1} = 0$) but not at the last position (implied by $\xi_h = 1$).

5.4 Computational Results

The proposed data-splitting algorithm is tested on various randomly generated instances comprising 30, 35 or 40 aircraft, under both arrival and mixed-mode traffic scenarios. The aircraft mixture for these instances is set as 40% Heavy, 40% Large, 20% Small. The target times of aircraft are randomly generated, with each aircraft appearing *Uniformly* once every γ seconds. Wider time-windows are considered where a flight can only arrive (or depart) up to 60 seconds earlier and no later than 1800 seconds past its scheduled target time. All of the computations are performed on a Windows machine, equipped with an Intel Xeon CPU E5-1630v3 3.70GHz 32GB RAM processor, using MATLAB and Gurobi 6.0.0 as the MIP-solver. The time-limit (TL) for computations is set as 7200 seconds for instance size of 30 or 35, and 18000 seconds for instance size of 40.

Multiple randomly generated instances for both arrival-only and mixed-traffic (50% arrival and 50% departure) scenarios are solved for $\gamma = 30$ or 35 seconds and $k \in 1, 2$. In order to demonstrate the effectiveness of optimizing the flight sequence, first capacity gain is provided in Table 5.2 which is observed to be 8% for $k = 1$, and 10.6% for $k = 2$ over FCFS. Next, as compared to the original 2-ASP formulation (5.1)-(5.11) (including variable fixing strategies), the proposed DS-ASP algorithm (Algorithm 5.2) results in over 78% reduction in the computational time required to solve all the randomly instances (Table 5.3), thereby providing tractable solutions. Observe that the mixed-traffic instances take more time to converge as the number of subset-pairs is large. Additionally, if the time-limit is raised, the percentage reduction could be more pronounced since as some of these instances may run for an indefinite times, unless time-limit is set. (Note that the average is taken over ten randomly generated instances for each k , and the range corresponds to the difference of minimum and maximum solver time obtained over ten instances.) This tremendous reduction in computational effort can be attributed to three reasons: (i) a significant reduction in search space;

(ii) a difference in solver behavior when solving large-scale versus small-scale instances, with the solver closing the gap more efficiently for smaller instances (verified by the consistency in solution time); and (iii) time taken to solve the LP-relaxation at each node is significantly lower in case of smaller instances.

Table 5.2 Effect of MPS parameter k on makespan averaged over 10 instances for each combination of instance size and γ , $k = 0$ denotes FCFS

F	γ	k			% Reduction (0-1)	% Reduction (0-2)
		0	1	2		
30	30	1435	1325	1290	7.7	10
35	30	1695	1564	1508	7.7	11
35	35	1683	1556	1503	7.5	10.7
40	30	1951	1786	1737	8.4	10.9
40	35	1929	1757	1731	8.9	10.3

Table 5.3 Computational time comparison between DS-ASP and 2-ASP with sequential implementation, where superscript a denotes arrival and m denotes mixed

F	$k \rightarrow$	Time (s)				% Reduction
		1		2		
	DS-ASP	2-ASP	DS-ASP	2-ASP		
	Avg. (Range)	Avg. (Range)	Avg. (Range)	Avg. (Range)		
30^a	30	159 (60 – 276)	2145 (529-5243)	539 (67-662)	> 6900 (4300– > 7200)	>92.3
35^a	30	326 (69 – 892)	> 7200 (> 7200– > 7200)	1543 (133 – 2657)	> 7200 (> 7200– > 7200)	>87
35^a	35	308 (170 – 794)	> 4752 (1287 – 7200)	1213 (614 – 1844)	> 6647 (1479– > 7200)	>86.6
40^a	30	412 (104 – 709)	> 18000 (> 18000– > 18000)	5417 (1151-12765)	> 18000 (> 18000– > 18000)	>83.8
40^a	35	657 (324 – 1209)	> 18000 (> 18000– > 18000)	4539 (548-10358)	> 18000 (> 18000– > 18000)	>85.5
40^m	30	795 (291 – 2229)	713 (130 – 1544)	9494 (557 – 19641)	> 15715 (4044– > 18000)	>37.3

DS-ASP is pleasingly parallel; each pair $(\ddot{A}, B)$ can be independently solved (without using information from any other pair), and finally solutions are compared to conclude optima. With parallelized implementation, as observed from Table 5.4, optima can be concluded in real or near-real time for almost all the instances.

Table 5.4 Computational time comparison between DS-ASP and 2-ASP with parallel implementation

F	$k \rightarrow$	Time (s)			
		1		2	
	γ	DS-ASP Avg. (Range)	2-ASP Avg. (Range)	DS-ASP Avg. (Range)	2-ASP Avg. (Range)
30^a	30	82 (52-123)	2145 (529-5243)	128 (65-200)	> 6900 (4300-> 7200)
35^a	30	141 (36-410)	> 7200 (> 7200-> 7200)	418 (69-1260)	> 7200 (> 7200-> 7200)
35^a	35	126 (58-317)	> 4752 (1287-7200)	385 (187-655)	> 6647 (1479-> 7200)
40^a	30	155 (74-294)	> 18000 (> 18000-> 18000)	1102 (401-1809)	> 18000 (> 18000-> 18000)
40^a	35	364 (152-713)	> 18000 (> 18000-> 18000)	980 (398-2561)	> 18000 (> 18000-> 18000)
40^m	30	136 (64-219)	713 (130-1544)	600 (204-1621)	> 15715 (892-> 18000)

Effect of γ

For the problem under consideration, inter-arrival time γ plays a key role in computational complexity. When time-separation matrix allows for average separation of the aircraft over both runways in the range of γ , *branch-and-cut* employed in commercial solver is able to rapidly close the gap between lower and upper bound, else a large number of nodes need to be explored during convergence. As a result, solution times increases with decreasing γ —rate of increase is exponential in case of 2-ASP thus driving it out of (near) real-time range. Experimental data recorded in Table 5.5 and Table 5.6 illustrate the effect of γ on computational time for both 2-ASP and DS-ASP. Observe that when γ is high, even large-sized instances, such as 40, can be solved instantaneously with direct use of a commercial solver. Computational complexity is a function of γ and instance size jointly; large-sized instances with lower γ are more complex on an average, e.g., instance of size 40 with γ equals to 30 is more complex than instance of size 40 with γ equals 40. Inevitably, peak period traffic instances entails large number of flights (instance size) with high frequency of movements (lower γ), thereby necessitating the use of this algorithm.

Table 5.5 Effect of γ on computation time for instance size 30 averaged over five instances for each γ

Mode of Operation	$k \rightarrow$	1		2	
	γ Time (s) $\searrow$	2-ASP	DS-ASP	2-ASP	DS-ASP
Arrival	40	4.81	-	92	-
Arrival	36	670	97	935	359
Arrival	34	751	126	> 3062	386
Arrival	32	803	100	> 4351	362
Arrival	30	1635	108	> 6567	567
Arrival	28	1862	107	> 6326	444

Table 5.6 Effect of γ on computation time for instance size 40 averaged over five instances for each γ

Mode of Operation	$k \rightarrow$	1		2	
	γ Time (s) $\searrow$	2-ASP	DS-ASP	2-ASP	DS-ASP
Arrival	50	0.42		3.21	
Arrival	44	241	215	963	829
Arrival	42	818	236	1061	934
Arrival	40	980	312	> 3042	1525
Arrival	38	> 1966	391	> 6060	1838

Effect of splitting point

The point of splitting plays a major role in the efficiency of DS-ASP. Combinatorially, symmetry leads to equal split; however, larger sized leading set leads to lower computational complexity as its aircraft are more sparsely spaced than that of the following set. Nonetheless, enlarging it beyond a certain point produces a reverse effect on the computations. Unfortunately, the optimal point of splitting cannot be decided deterministically at the onset as it is the result of an interplay of many factors, including γ . Therefore, empirical analysis is used and it is observed that in general, for any value of γ , the cardinality of the following set should not exceed 40% of entire set or 15, whichever is lower, as the following set large than 15 is computationally expensive. (The same rule is followed for the computations in Table 5.5-5.7). Further, computations recorded in Fig 5.8 and Fig 5.9 illustrate this effect for each combination of γ and splitting point (averaged over five instances) of arrival traffic with instance size 30. It can be observed that for both values of k , efficiency increases with the size of the leading set and then starts to decrease after attaining a peak.

Fig. 5.8 Effect of splitting on computation time for arrival traffic of size 30 in case of $k = 1$ (IAT represents γ)

Fig. 5.9 Effect of splitting on computation time for arrival traffic of size 30 in case of $k = 2$

5.5 Summary

In this chapter, the aircraft sequencing problem is solved to optimally allocate and schedule the aircraft on dual-independent runways under both arrival only and mixed traffic conditions, respecting safety, equity, and time-window constraints, while maximizing the throughput of the system. Recognizing the fallibility of directly using a 0–1 MIP approach and limitations of the existing algorithms and solvers to provide real-time optimal solutions for large scale realistic instances of this problem, a novel solution based on data-splitting algorithm is proposed. Framework for multiple split is also presented to very large-size instances. Perfor-

mance of the DS-ASP algorithm clearly demonstrates the computational speed-up obtained, while concluding optimality in all the practical test-bed instances; sequential implementation can solve in real or near-real time for maximum position shifting parameter equal one, while parallelized implementation can solve nearly all of the instances in real or near-real time for value of this parameter up to two.

Next chapter extends the ASP formulation to include constraints along entire TMA region recognizing that bottleneck may appear anywhere within the TMA, and not just on runway. Further, it also presents the computations to illustrate the benefit of extending the ASP to entire TMA.

Chapter 6

Aircraft sequencing over entire TMA

6.1 Introduction

Recognizing that terminal area scheduling is much more useful from an airport's point of view, while also affording greater flexibility in managing aircraft, in this chapter, the sequence of arriving and departing aircraft is optimized and optimal flight schedules, based on the entire *Terminal Manoeuvring Area (TMA)*, are determined, with the objective of minimizing total delay in the system.

As shown in Chapter 2 (Fig 2.3), the TMA is a circular region with radius of approximately 50NM centered at the airport. At the boundary of the TMA, there are various *way-points* or *fixes*, labeled as entry (or meter) gates, through which traffic enters the TMA. The traffic from the enroute airspace merges at each entry gate and flows into the TMA as *one stream* from each gate. Given a stream, there is a pre-specified route leading to the runway (a route can be best defined as a series of way-points) and where the sections of a route between two consecutive way-points is said to form a *trajectory segment*. Multiple routes may share one or more such trajectory segments. An incoming aircraft from enroute airspace either enters into the holding area near the entry gates or proceeds directly along a pre-specified route to its allocated runway. Traffic from different streams merges together at common way-points (merge points) and subsequently flows along the same route. Finally, all traffic designated for the same runway merges into a single stream at the *final approach fix (FAF)* and descends into the runway

threshold. (This operation is normally performed at a distance of 5-10NM from the runway.) Similarly, for departures, aircraft flow together towards the *initial fix* near the runway, and then diverge along different trajectory segments at way-points, finally exiting the TMA from the exit gates.

Once an approach procedure is initiated at the entry gates, a switch in position between aircraft on the same route is not permitted; furthermore, during this entire approach operation, safety separation must be maintained between consecutive and non-consecutive pairs of aircraft. In contrast to the time-based separation standards enforced on the runway, the FAA-specified longitudinal safety separation requirements in the TMA region for both arrivals and departures are *distance-based*, which are shown in Table 6.1 below.

Table 6.1 Distance-based TMA separation standards (in nautical miles) enforced for arrivals

Leading / Following	Heavy	Large	Small
Heavy	4	5	6
Large	3	3	4
Small	3	3	3

The flight sequencing and scheduling operations along the TMA are modeled as 0-1 MIP, which is an adaptation of Beasley et al. (2000)'s model enhanced with the inclusion of the safety separation along the entire TMA, CPS constraints, and a modified objective of minimizing the total delay in the system. The approach is a holistic approach (addressing safety, efficiency, and equity) that presents manifold improvements as compared to earlier works, both from a modeling perspective as well as practical considerations, such as:

- (i) Extension of the ASP to incorporate the configuration of the TMA, thereby accounting for all safety constraints and bottlenecks in the system;
- (ii) Inclusion of the CPS constraint within the optimization formulation, which forms an important component in maintaining equity;
- (iii) This formulation also yields the advantage of determining exact solutions, as compared to heuristics implementation often seen in practice, thereby improving overall system efficiency.

The remainder of this chapter is organized as follows. In Section 6.2, the aircraft sequencing problem along the entire TMA is formulated, and several variable fixing strategies are proposed. Computational results illustrating the benefit of TMA sequencing over runway, along with some preliminary computations

that serve to highlight the inherent difficulty involved in solving this problem via a straightforward MIP approach, are presented in Section 6.3, and finally, Section 6.4 summarizes the contributions of this work.

6.2 Modeling and Analysis

As shown in Fig. 6.1, the TMA region has been represented as a *graph*, where the waypoints are shown as *nodes* or (*vertices*) and the trajectory segments between two consecutive waypoints as *edges*. Arrival traffic enters into the TMA through various entry gates and departure traffic enters into the runway through a holding area, which is adjacent to the runway. It is well-established that absorbing anticipated delays at higher altitudes, is better from fuel consumption, safety, and controller workload viewpoints (Erzberger 1995). Hence, in this model representation, it is assumed that delay or earliness is induced in an aircraft before entering the TMA, either by adjusting its velocity or by holding at the entry gates. (For departure traffic, delay is induced on the ground.) As aforementioned, when compiling the final runway sequence, the nominal order of aircraft in the same stream is always preserved, thereby ensuring that overtaking is avoided and a runway closest to the entry fix of the aircraft is allocated.

Fig. 6.1 A graphical representation of a TMA region with four entry fixes, four merge points, and a single runway

The aircraft sequencing problem defined over the TMA, called ASP-TMA is now formulated next, where the definition of the index sets, parameters, and decision variables is as follows.

Description of Index Sets and Parameters

- $n \in \mathcal{N} \equiv \{1, 2, \dots, N\}$: Set of all nodes in the graph representing entry gates, merge points, departure holding area, and the runway
- $g \in \mathcal{G}$: Set of all the nodes in the graph representing entry gates, where $\mathcal{G} \subseteq \mathcal{N}$
- $(p, q) \equiv e \in \mathcal{E}$: Set of all the edges (trajectory segments) in the graph, where $p \in \mathcal{N}$, $q \in \mathcal{N}$
- r : Vertex representing runway
- $f \in \mathcal{F}$: Set of all arriving and departing flights
- v_i^e : Velocity of any flight i in trajectory segment $e \in \mathcal{E}$
- $\mathcal{F}^n$: Set of flights passing through vertex n , where $\mathcal{F}^n \subseteq \mathcal{F}$
- E_i^n : Earliest time of arrival (departure) of aircraft i at (from) vertex n
- T_i^n : Target time of arrival (departure) of aircraft i at (from) vertex n
- L_i^n : Latest arrival (departure) of any aircraft i at (from) vertex n
- Δt_{ij} : Minimum safety separation (in seconds) at runway threshold if flight i is ahead of flight j
- Δs_{ij} : Minimum safety separation (in miles), if flight i is ahead of flight j (see Table 1.1)
- seq_i : Position of flight i based on the FCFS sequence
- k : Specified maximum position shifting (MPS) parameter

Decision Variables

- $x_{ij}^n = \begin{cases} 1, & \text{if flight } i \text{ is ahead of flight } j \text{ in schedule at vertex } n \\ 0, & \text{otherwise} \end{cases}$
- t_i^n = Scheduled time of arrival (departure) of flight i at (from) vertex n

$$\text{ASP-TMA:} \quad \text{Minimize } \sum_{f \in \mathcal{F}} |t_f^r - T_f^r| \quad (6.1)$$

$$\text{subject to } x_{ij}^n + x_{ji}^n = 1, \quad \forall i < j, (i, j) \in \mathcal{F}^n, \forall n \in \mathcal{N} \quad (6.2)$$

$$t_i^q = t_i^p + T_i^q - T_i^p, \quad \forall i \in \mathcal{F}^p \cap \mathcal{F}^q, \forall (p, q) \in \mathcal{E} \quad (6.3)$$

$$t_j^p \geq t_i^p + (\Delta s_{ij}/v_i^e) - M(1 - x_{ij}^p), \quad \forall i, j \in \mathcal{F}^p \quad (6.4)$$

$$t_j^q \geq t_i^q + (\Delta s_{ij}/v_j^e) - M(1 - x_{ij}^q), \quad \forall i, j \in \mathcal{F}^q, \forall (p, q) \in \mathcal{E} \quad (6.5)$$

$$t_j^r \geq \Delta t_{ij} + t_i^r - M(1 - x_{ij}^r), \quad \forall i, j \in \mathcal{F} \quad (6.6)$$

$$-k \leq (F - \sum_{j \in \mathcal{F}, i \neq j} (x_{ij}^r)) - seq_i \leq k, \quad \forall i \in \mathcal{F} \quad (6.7)$$

$$x_{ij}^q = x_{ij}^p \quad \forall i < j, (i, j) \in \mathcal{F}^p, \forall (p, q) \in \mathcal{E} \quad (6.8)$$

$$E_n^i \leq t_n^i \leq L_n^i \quad \forall i \in \mathcal{F}^n, \forall n \in \mathcal{N} \quad (6.9)$$

$$x_{ij}^n \in \{0, 1\}, \quad t_i^n \geq 0 \quad \forall i, j \in \mathcal{F}^n, \forall n \in \mathcal{N} \quad (6.10)$$

In the above formulation, the objective function (6.1) seeks to minimize the total delay; constraint (6.2) enforces the order precedence between flights i and j at every vertex; constraint (6.3) computes the scheduled time of arrival (or departure) of flight i arriving at node q from node p , where $(p, q) \in \mathcal{E}$, based on the condition of free flow of flights once they enter the TMA; constraint (6.4) ensures

the longitudinal separation requirement (in miles) between flights departing from each $n \in \mathcal{N} \setminus \{r\}$ (Fig 6.2a); constraint (6.5) ensures the longitudinal separation requirement (in miles) between flights arriving at each $n \in \mathcal{N} \setminus \{r\}$ (Fig 6.2b). Note that the preceding two constraints will ensure that safety separation is not violated throughout the trajectory.

(a) Flights i and j departing from a node

(b) Flights i and j arriving at a node

Fig. 6.2 In figure 6.2a, when flight j reaches the node p , flight i must be at a safety separation of Δs_{ij} distance units or $\frac{\Delta s_{ij}}{v_i^j}$ time units. In figure 6.2b, when the leading flight i reaches the node q , flight j must be Δs_{ij} distance units behind or $\frac{\Delta s_{ij}}{v_j^i}$ time units.

Constraint (6.6) ensures the time-based separation requirement between flights at the runway node r ; constraint (6.7) imposes the CPS constraint that an aircraft cannot be shifted by more than k positions from its initial (FCFS) position, where $F \equiv |\mathcal{F}|$; constraint (6.8) avoids overtaking by maintaining the precedence relationship between two aircraft at succeeding way points; constraint (6.9) maintains the scheduled time of arrival (departure) at each vertex to be between the earliest and latest times for each aircraft; and finally constraint (6.10) imposes the binary and non-negativity restrictions on the x - and t -variables, respectively.

The ASP-TMA formulation described above can be further enhanced by prefixing some of the variables by using the variable fixing strategies of Chapter 3, after customizing them to the present context.

Variable Fixing Strategies

Proposition 6.1. *At any vertex n , consider a pair of aircraft $i, j \in \mathcal{F}^n$. If the FCFS sequence positions of i and j satisfy the condition: $\text{seq}(j) - \text{seq}(i) \geq 2k$, where k is the MPS parameter, then $x_{ij}^n = 1$ is fixed.*

Proposition 6.2. *At any entry waypoint g , consider a pair of aircraft $i, j \in \mathcal{F}^g$ that belong to the same category. Suppose that the time-window restrictions*

satisfy the following conditions:

(i) $E_i^g \leq E_j^g$; (ii) $T_i^g \leq T_j^g$ and (iii) $L_i^g \leq L_j^g$. Then, x_{ij}^g can be set to 1. (It should be noted that this prefixing routine cannot be applied at merge points as flights originating at different entry fixes are in negotiation at each intermediate node.)

6.3 Computational Results

For the sample TMA configuration shown in Fig. 6.1, a set of 10 aircraft belonging to different weight categories, and comprising eight arrivals and two departures is considered. The earliest time of arrival (or departure) for all aircraft is set as 60 seconds before their respective target time and the latest time is taken to be one hour beyond the target time. The velocity of heavy, large and small categories of aircraft is set at 260nm/hr, 160nm/hr, and 140nm/hr, respectively, and is assumed to be constant throughout the TMA. Arriving flights either stack up at entry gates (labeled as nodes $\{1,2,3,4\}$), or enter the TMA, merging with other streams at merge points (labeled as nodes $\{6,7,8\}$), before finally merging with scheduled departures (at node labelled $\{9\}$), resulting in the final runway sequence.

We test the relative effectiveness of the schedule generated by three different sequences: (i) optimal runway sequence (obtained by ignoring the TMA configuration constraints and enforcing only the runway separation standard in the ASP-TMA formulation); (ii) FCFS, and (iii) the sequence obtained using the ASP-TMA formulation. Fig. 6.3 and Fig. 6.4 display the optimal sequences and associated delays obtained using the aforementioned sequencing algorithms, corresponding to the MPS parameter $k = 1$ and $k = 3$.

As seen in Fig. 6.3, allowing an aircraft to shift its position by even one unit with respect to its FCFS sequence-based position can result in a significant reduction in total delay, and the accrued delay reduction increase with an increase in the value of k (Fig. 6.4). Further, it can be observed that the third sequence results in extra delay reduction of 15% and 27% when compared to the first sequence in case of $k = 1$ and $k = 3$ respectively, thereby reinforcing the effectiveness of the proposed ASP-TMA problem.

Table 6.2 records the CPU times when the ASP-TMA (along with variable fixing strategies outlined in Propositions 1-2) is directly using a commercial solver, on randomly generated instances of arrival traffic with different instance size. It

Fig. 6.3 Optimal sequences obtained by the three sequencing models for the case of $k = 1$.

Fig. 6.4 Optimal sequences obtained by the three sequencing models for the case of $k = 3$.

can be observed that even though the above mentioned variable fixing strategies speed up the solution process, the solver still faces the computational difficulty in handling large scale instances, particularly for $k \geq 2$, thereby warranting the development of an efficient algorithm.

Table 6.2 Computational times for large-scale realistic instances on direct implementation of ASP-TMA

Instance	# of flights	MPS (k)		
		1	2	3
1	20	10	21	44
2	20	10	23	56
3	30	24	69	3206
4	30	23	52	374
5	40	46	320	> 3600
6	40	45	1497	> 3600

6.4 Summary

In this chapter, the aircraft sequencing problem over the entire TMA region is posed and solved, taking into account several realistic constraints, such as longitudinal separations at the runway and in the TMA airspace, CPS constraints, and precedence maintenance requirements, with the objective of minimizing the total delay in the system. This complex problem is formulated as a 0-1 MILP, and variable fixing strategies are used to improve the computational efficiency of the model. The results indicate that significant delay savings can be achieved over the base FCFS policy schedule and over the runway optimized sequence. The model is generic enough to be applied to different types of TMA configurations and can handle any mixture of traffic for the single runway case under segregated or mixed-modes of operations. However, computational difficulty in exactly solving the large scale instances affirm the need of the algorithmic development to generate high quality solution in real-time, which is one of the important directions of future research.

The next chapter summarizes the findings and presents direction for future extension.

Chapter 7

Conclusions

7.1 Summary

This research investigated aircraft sequencing problem with special focus on obtaining tractable and robust solutions for its important variants. The objective of this research was to optimize runway capacity, maximize airport performance, as well as reduce congestion in terminal maneuvering area, which are often correlated goals, while satisfying the constraints on safety and equity. The challenge was to solve large scale realistic instances of this NP-hard problem to optimality, in real-time as airport environment is very dynamic and uncertain, where data keeps getting updated constantly and necessitate fast sequencing or resequencing decisions.

The detailed study of ASP paved the way to development of Data-Splitting Algorithm, called DS-ASP, a novel solution technique which tractably solves many important variants; is pleasingly parallelizable, provide fast and scalable computations for large scale realistic instances, and can be tailored to account for several other problem variants. This algorithm is based on divide-and-conquer strategy wherein the given set of ights is divided into several subsets, each of which is then optimized using 0–1 MIP; the novelty is introduced in both dividing and solving so as to ensure the optimality of the entire set.

For the first time, an LP-based branch-and-cut approach is able to tractably solve the peak-traffic instances of ASP in presence of CPS constraints. This includes solving for a very difficult and important objective of minimizing delay in presence of time-windows which has not been solved so far even by an exist-

ing state-of-the-art algorithm based on dynamic programming, a very effective method for several other objective functions. Further, DS-ASP also efficiently solves for the objective function of maximizing throughput, including the scenario of mixed-mode operations and $k \geq 4$, for which dynamic programming suffers the curse of dimensionality. The most important contribution of this work is to solve dual independent runway sequencing and scheduling problem in presence of constrained position shifting; no algorithm thus far can tractably solve this problem which is very complex due to the extra runway allocation decision that has to be made.

Finally, an optimization framework for the terminal area scheduling is presented which captures safety, efficiency and equity for the first time in literature, while accounting for all the important constraints. Computation on a specific test instance illustrates significant delay savings over the scheduling procedure which only takes runway constraints into account, thereby making it a promising area for future research both from the modeling as well as algorithmic perspective.

Computations on numerous randomly generated instances demonstrate significant reduction in the solution time over existing methods and directly using commercial solvers, and remarkable speedup, thus pointing the use of these algorithms in arrival-departure management systems which today mostly rely on heuristics for real-time sequencing and scheduling. In case of ASP on a single runway with the objective function of minimizing delay, the average reduction is found to be as much as 99% in getting optimal solution when compared to a commercial solver; the reduction is as much as 99% in case of objective function of maximizing throughput; and, finally in case of dual runways sequencing and scheduling, the reduction is observed to be as much as 91%. The reductions are more pronounced when algorithm is implemented in parallel.

7.2 Future Research Opportunities

The work in this thesis can lead to several research directions mentioned below.

7.2.1 Enhanced TMA sequencing and scheduling

For the TMA model formulated in Chapter 6, computational experiments reveal some deficiencies in getting real-time optimal solutions for large-sized

instances, under early landings and wide time-windows, when directly solved with a commercial MIP solver. This necessitates further model improvements and development of specialized algorithm in order to reduce the computational efforts. Development of a metaheuristic which can address this problem comprehensively or an efficient optimal solution was not in the scope of this thesis; nonetheless, it constitutes a good research question. An another important direction for future research pertaining to this could be to improve the TMA representation; in this work, a very generic representation of TMA is presented wherein delay is absorbed only at the holding points at the entry gate. However, there can be airport specific delay absorption structures such as point merge, and holding inside TMA which can be accounted in the formulation.

7.2.2 Incorporating other operational constraints and objective functions

In the dynamic and uncertain airport environment consisting of multiple stakeholders, there can be several other objective functions and operational constraints which have not been considered in this dissertation. For example, other objective functions those can be considered include the minimizing weighted delay, minimizing operating cost, etc., or minimizing delay in case of dual runway system. Further, constraints can be imposed on number of heavy arrivals or total arrivals on a particular runway, in dual runway system. Inclusion of each objective function or operational constraints may affect the problem structure, and hence customizing DS-ASP to obtain optimal solution while satisfying those conditions can be an interesting direction of research.

7.2.3 Extending the DS-ASP for multiobjective ASP

This dissertation and most of the work in literature has mainly focused on single objective optimization. However, multiple objectives satisfying diverse stakeholders may need to be considered. It would be interesting to model these objectives together, and investigate the application of DS-ASP so as to reap its computational advantage.

7.2.4 Enhancing the performance of DS-ASP

Solving leading and following set independently in case of data-splitting algorithm offers an avenue to improve its performance. The possible directions to explore are pruning more instance pairs, design performance enhancement routines by delving deeper into the data, and the efficient algorithmic implementation. These improvements has direct implications for the very complex scheduling problems such as dual independent runway scheduling.

7.2.5 Extending the DS-ASP to other scheduling problems

ASP is equivalent to ATSP, and therefore there is an opportunity to apply data splitting algorithm to other scheduling problems rooted in ATSP, particularly with constrained position shifting requirements. Further, the insight and analysis presented in this dissertation can be used to design other novel solutions for these problems. Example of such problems are job shop scheduling problems, vehicle re-sequencing problems in automotive assembly, and vehicle routing problems with dynamic requests for point-to-point transportation.

7.2.6 Comparison of DS-ASP with metaheuristics

Even though metaheuristics have not been touched upon in this thesis as its focus was the development of exact algorithms which can solve peak-traffic instances of reasonable size in real-time, metaheuristics can be equally good in generating high quality solutions. As part of the future research opportunities, it would be interesting to research on a suitable metaheuristics and compare the performance with DS-ASP for each problem variant, both in terms of solution quality and solution time, particularly in the case of very large-sized peak traffic instances.

References

- Abela, J., Abramson, D., Krishnamoorthy, M., De Silva, A., and Mills, G. (1993). Computing optimal schedules for landing aircraft. In *Proceedings of the 12th National ASOR Conference*. Adelaide, Australia.
- Airbus (2016). *Global Market Forecast 2016-2035*. http://www.airbus.com.cn/fileadmin/media_gallery/files/brochures_publications/GMF/Global_Market_Forecast_2016-2035.pdf.
- Airlines for America (2017). U.S. passenger carrier delay costs. Accessed December 2017, <http://airlines.org/dataset/per-minute-cost-of-delays-to-u-s-airlines/>.
- Anagnostakis, I., Clarke, J.-P., Böhme, D., and Völcker, U. (2001). Runway operations planning and control: Sequencing and scheduling. *Journal of Aircraft*, 38(6):988–996.
- Arkind, K. D. (2004). Requirements for a novel terminal area capacity enhancement concept in 2022. *AIAA Paper*, 5411.
- Ascheuer, N., Fischetti, M., and Grötschel, M. (2001). Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90(3):475–506.
- Atkin, J. A., Burke, E. K., Greenwood, J. S., and Reeson, D. (2007). Hybrid metaheuristics to aid runway scheduling at London Heathrow Airport. *Transportation Science*, 41(1):90–106.
- Avella, P., Boccia, M., Mannino, C., and Vasilyev, I. (2017). Time-indexed formulations for the runway scheduling problem. *Transportation Science*, 51(4):1196–1209.
- Balakrishnan, H. and Chandran, B. G. (2010). Algorithms for scheduling runway operations under constrained position shifting. *Operations Research*, 58(6):1650–1665.
- Balas, E. and Simonetti, N. (2001). Linear time dynamic-programming algorithms for new classes of restricted tsps: A computational study. *INFORMS Journal on Computing*, 13(1):56–75.
- Baldacci, R., Maniezzo, V., and Mingozzi, A. (2004). An exact method for the car pooling problem based on lagrangean column generation. *Operations Research*, 52(3):422–439.

- Ball, M., Barnhart, C., Dresner, M., Hansen, M., Neels, K., Odoni, A., Peterson, E., Sherry, L., Trani, A., Zou, B., et al. (2010). Total delay impact study. Technical report, NEXTOR.
- Beasley, J., Sonander, J., and Havelock, P. (2001). Scheduling aircraft landings at London Heathrow using a population heuristic. *The Journal of the Operational Research Society*, 52(5):483–493.
- Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D. (2000). Scheduling aircraft landings - The static case. *Transportation Science*, 34(2):180–197.
- Bennell, J. A., Mesgarpour, M., and Potts, C. N. (2011). Airport runway scheduling. *4OR: A Quarterly Journal of Operations Research*, 9(2):115–138.
- Bianco, L., Dell’Olmo, P., and Giordani, S. (1999). Minimizing total completion time subject to release dates and sequence-dependent processing times. *Annals of Operations Research*, 86:393–415.
- Bianco, L., Dell’Olmo, P., and Giordani, S. (2006). Scheduling models for air traffic control in terminal areas. *Journal of Scheduling*, 9(3):223–253.
- Bianco, L., Dell’Olmo, P., and Odoni, A. R. (1997). *Modelling and Simulation in Air Traffic Management*. Springer, Berlin.
- Briskorn, D. and Stolletz, R. (2014). Aircraft landing problems with aircraft classes. *Journal of Scheduling*, 17(1):31–45.
- Carr, F. R. (2004). *Robust decision-support tools for airport surface traffic*. PhD thesis, Cambridge, MA.
- D’Ariano, A., Pacciarelli, D., Pistelli, M., and Pranzo, M. (2015). Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area. *Networks*, 65(3):212–227.
- Davis, T. J., Krzeczowski, K., and Bergh, C. (1994). The final approach spacing tool. In *Proceedings of the 13th IFAC Symposium on Automatic Control in Aerospace*. Pao Alto, CA.
- Dear, R. G. (1976). The dynamic scheduling of aircraft in the near terminal area. Technical report, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA.
- Desai, J. and Prakash, R. (2016). An optimization framework for terminal sequencing and scheduling: The single runway case. In *Proceedings of the Complex Systems Design & Management 2016 Asia Conference*, pages 195–207. Springer, Cham.
- Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354.

- Ernst, A. T., Krishnamoorthy, M., and Storer, R. H. (1999). Heuristic and exact algorithms for scheduling aircraft landings. *Networks*, 34(3):229–241.
- Erzberger, H. (1995). Design principles and algorithms for automated air traffic management. *Knowledge-Based Functions in Aerospace Systems*, 7(2).
- Federal Aviation Administration (2018). Accessed January 2018, <https://aspm.faa.gov/opsnet/sys/main.asp>.
- Furini, F., Kidd, M. P., Persiani, C. A., and Toth, P. (2014). State space reduced dynamic programming for the aircraft sequencing problem with constrained position shifting. In *Combinatorial Optimization: Third International Symposium*, pages 267–279. Springer, Cham.
- Furini, F., Persiani, C. A., and Toth, P. (2012). Aircraft sequencing problems via a rolling horizon algorithm. In *Combinatorial Optimization: Second International Symposium*, pages 273–284. Springer, Berlin.
- Garcia, J.-L. (1990). Maestro- a metering and spacing tool. In *Proceedings of the American Control Conference*, pages 502–507. San Diego, CA.
- Ghoniem, A., Farhadi, F., and Reihaneh, M. (2015). An accelerated branch-and-price algorithm for multiple-runway aircraft sequencing problems. *European Journal of Operational Research*, 246(1):34–43.
- Ghoniem, A., Sherali, H. D., and Baik, H. (2014). Enhanced models for a mixed arrival-departure aircraft sequencing problem. *INFORMS Journal on Computing*, 26(3):514–530.
- Hansen, J. V. (2004). Genetic search methods in air traffic control. *Computers & Operations Research*, 31(3):445–459.
- Harikiopoulou, D. and Neogi, N. (2011). Polynomial-time feasibility condition for multiclass aircraft sequencing on a single-runway Airport. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):2–14.
- Hu, X.-B. and Chen, W.-H. (2005a). Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Engineering Applications of Artificial Intelligence*, 18(5):633–642.
- Hu, X.-B. and Chen, W.-H. (2005b). Receding horizon control for aircraft arrival sequencing and scheduling. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):189–197.
- Idris, H., Delcaire, B., Anagnostakis, I., Hall, W., Pujet, N., Feron, E., Hansman, R., Clarke, J.-P., and Odoni, A. (1998). Identification of flow constraint and control points in departure operations at airport systems. In *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*, page 4291. American Institute of Aeronautics and Astronautics, Reston, VA.
- Kallehauge, B., Larsen, J., Madsen, O. B., and Solomon, M. M. (2005). Vehicle routing problem with time windows. *Column generation*, pages 67–98.

- Krafft, O. and Schaefer, M. (2002). On the number of permutations within a given distance. *Fibonacci Quarterly*, 40(5):429–434.
- Lahmar, M. and Benjaafar, S. (2007). Sequencing with limited flexibility. *IIE Transactions*, 39(10):937–955.
- Lee, H. and Balakrishnan, H. (2008). A study of tradeoffs in scheduling terminal-area operations. *Proceedings of the IEEE*, 96(12):2081–2095.
- Lieder, A., Briskorn, D., and Stolletz, R. (2015). A dynamic programming approach for the aircraft landing problem with aircraft classes. *European Journal of Operational Research*, 243(1):61–69.
- Lieder, A. and Stolletz, R. (2016). Scheduling aircraft take-offs and landings on interdependent and heterogeneous runways. *Transportation research part E: logistics and transportation review*, 88:167–188.
- Neuman, F. and Erzberger, H. (1991). *Analysis of delay reducing and fuel saving sequencing and spacing algorithms for arrival traffic*. NASA Ames Research Center, Moffett Field, CA.
- Pinol, H. and Beasley, J. E. (2006). Scatter search and bionomic algorithms for the aircraft landing problem. *European Journal of Operational Research*, 171(2):439–462.
- Psaraftis, H. N. (1978). *A dynamic programming approach to the aircraft sequencing problem*. PhD thesis, Massachusetts Institute of Technology, Cambridge.
- Psaraftis, H. N. (1980). A dynamic programming approach for sequencing groups of identical jobs. *Operations Research*, 28(6):1347–1359.
- Samà, M., D’Ariano, A., D’Ariano, P., and Pacciarelli, D. (2014). Optimal aircraft scheduling and routing at a terminal control area during disturbances. *Transportation Research Part C: Emerging Technologies*, 47:61–85.
- Sherali, H. D. and Adams, W. P. (1990). A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430.
- Sherali, H. D. and Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems. *Discrete Applied Mathematics*, 52(1):83–106.
- Sölveling, G. and Clarke, J.-P. (2014). Scheduling of airport runway operations using stochastic branch and bound methods. *Transportation Research Part C: Emerging Technologies*, 45:119–137.
- Soomer, M. and Franx, G. J. (2008). Scheduling aircraft landings using airlines preferences. *European Journal of Operational Research*, 190(1):277–291.

- Volckers, U. (1990). Arrival planning and sequencing with COMPAS-OP at the Frankfurt ATC-Center. In *Proceedings of the 1990 American Control Conference*, pages 496–501. San Diego, CA.
- Wen, M., Larsen, J., and Clausen, J. (2005). An exact algorithm for aircraft landing problem. Technical report, Technical University of Denmark, Lyngby, Denmark.
- Wong, G. L. (2000). The dynamic planer: the sequencer, scheduler, and runway allocator for air traffic control automation. Report NASA TM-2000-209586, NASA Ames Research Center.
- Zhan, Z.-H., Zhang, J., Li, Y., Liu, O., Kwok, S., Ip, W., and Kaynak, O. (2010). An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):399–412.

Appendix A

Time-windows Tightening

E_i (resp. L_i) values for any flight i can be tightened to give tighter early (resp. late) time restrictions, say E_i^{new} (resp. L_i^{new}), where $E_i^{new} \geq E_i$ (resp. $L_i^{new} \leq L_i$). The following optimization problems are now formulated by defining two new sets of aircraft, $\mathcal{F}_i^+ = \{j : x_{ji} = 1\}$ (resp. $\mathcal{F}_i^- = \{j : x_{ij} = 1\}$), which are known in advance to land before (resp. after) aircraft i (after applying variable fixing strategies).

$$\begin{aligned}
 E_i^{new} &\equiv \text{Minimize } t_i & L_i^{new} &\equiv \text{Maximize } t_i \\
 \text{s.t } x_{ji} &= 1 \quad \forall j \in \mathcal{F}_i^+ & \text{s.t } x_{ij} &= 1 \quad \forall j \in \mathcal{F}_i^- \\
 t_i &\geq t_j + \Delta_{ji} \quad \forall j \in \mathcal{F}_i^+ & t_j &\geq t_i + \Delta_{ij} \quad \forall j \in \mathcal{F}_i^- \\
 \{\text{ASP} : 1-6\} &\quad \forall j \in \mathcal{F}_i^+ & \{\text{ASP} : 1-6\} &\quad \forall j \in \mathcal{F}_i^-
 \end{aligned} \tag{A.1}$$

Even though the above time-windows tightening formulation is found to be very effective, computationally it is expensive. Therefore, a simple heuristics is proposed based on an analysis done by Psaraftis (1978) that can tighten the time-windows in an efficient manner. According to Psaraftis (1978), group clustering takes place for arrival traffic in case of the objective of maximizing throughput traffic, if CPS requirements and time-windows restrictions are lifted, and all the flights are ready to land at the same time. In the optimal sequence resulting from group clustering, small arrivals land first, followed by large arrivals, which are then followed by heavy arrivals. The whole process can be outlined in the form of the following pseudo-code:

Algorithm A.1: Heuristic for generating tighter time-windows

- 1: Set $\mathcal{P} \leftarrow \mathcal{F}$.
 - 2: If $\mathcal{P} = \emptyset$, go to Step 6; else, select any flight i from $\mathcal{P}$, update $\mathcal{P} \leftarrow \mathcal{P} \setminus \{i\}$.
 - 3: Apply variable fixing strategies to get F_i^+ and F_i^- .
 - 4: Relax time-windows and CPS requirements, and then, using the group clustering method, calculate the makespan z^+ and z^- for sets $\mathcal{F}_i^+ \cup \{i\}$ and $\mathcal{F}_i^- \cup \{i\}$, respectively.
 - 5: Calculate: $E_i^{new} = \max\{E_i, E_1 + z^+\}$ and $L_i^{new} = \min\{L_i, L_n - z^-\}$, where $n = |F|$. Go to Step 2.
 - 6: Update: $E_i = E_i^{new}, L_i = L_i^{new} \forall i \in \mathcal{F}$. STOP.
-

Appendix B

Search Space Reduction

B.1 Combinatorial Exposition of Search Space Reduction

The motivation in developing the DS-ASP is to reduce the search space significantly by decomposing the overall data set into two smaller sets, with each of these smaller data sets being independently optimized. This leads to an appreciable reduction in the computational time required to find an optimal solution. In the following discussion, we present a combinatorial exposition of the extent of search space reduction is presented; the number of flight sequences is also shown to be the least when the cardinality of the two smaller data-sets, namely, A and B , is the same, both in the absence, and presence, of CPS constraints.

Consider a set of aircraft of size n , which is split into two subsets of sizes m and $n - m$. Let I_m denote the resulting flight sequences (permutations) after performing data-splitting. It is referred to the total number of possible permutations that result after fixing the last aircraft in the first subset (refer to Section 3.3 for further details). Clearly, when $m = n$, i.e., in the absence of data-splitting, I_n reduces to the total number of permutations in the overall ASP formulation.

Now, consider the following two cases:

Case 1. (No CPS constraint) In the absence of data splitting, considering all n flights simultaneously, the total number of permutations clearly is $n!$. After data-splitting and generating the instance pairs, the size of the search space becomes:

$$I_m = \binom{n}{m} m[(m-1)! + (n-m)!], \quad 3 \leq m \leq n-2,$$

which is clearly less than $n!$. For $3 \leq m \leq n - 2$ to be satisfied, n being an even number (to allow for the data-set to be split into two equal halves), n should be ≥ 6 flights. Indeed, for values of $n \leq 5$, ASP is a trivial problem, and an optimal solution can be easily determined through a brute force enumeration strategy.

Now, consider the following conditions on m :

Case 1.1. (Equal Splitting) Substituting $m = \frac{n}{2}$, and with some algebraic manipulations:

$$I_{\frac{n}{2}} = (\frac{n}{2} + 1)(\frac{n}{2} + 1)(\frac{n}{2} + 2)(\frac{n}{2} + 3) \cdots (n) < n!$$

Case 1.2. (Unequal Splitting) If $m < \frac{n}{2}$, then,

$$\begin{aligned} I_m &= \binom{n}{m} m[(m-1)! + (n-m)!] \\ &= \binom{n}{m} [m! + m(n-m)!] > m \binom{n!}{m!} \\ &= (m)(m+1)(m+2) \cdots (\frac{n}{2})(\frac{n}{2} + 1) \cdots (n) > I_{\frac{n}{2}}. \end{aligned}$$

Case 2. (With CPS constraint) For simplicity of exposition and ease of notation, consider the case of $k = 1$. When $k = 1$, the number of pairs $(\dot{A}, B)$ can at most be three, where the last aircraft of $\dot{A}$ in the first pair, last two aircraft of $\dot{A}$ in the second pair, and, the last aircraft of $\dot{A}$ and the first aircraft of B in the third pair, will not change their positions in their corresponding optimal sequences. Based on this observation, the number of arrangements for the three listed pairs are $f_{m-1}+f_{n-m}$, $f_{m-2}+f_{n-m}$, and $f_{m-1}+f_{n-m-1}$, respectively, where f_i denotes the number of permutations of an i -sized sequence of aircraft. From Krafft and Schaefer (2002), the number of permutations, f_i , satisfy the relation $f_i = f_{i-1} + f_{i-2}$, i.e., the number of permutations follow the Fibonacci sequence, and therefore, the total number of permutations in this case is given by $f_{m+1} + f_{n-m+2}$.

Now, consider the following conditions on m :

Case 2.1. $m = \frac{n}{2}$. In this case,

$$I_{\frac{n}{2}} = f_{\frac{n}{2}+1} + f_{n-\frac{n}{2}+2} = f_{\frac{n}{2}+3} < f_n \equiv I_n.$$

Case 2.2. $m < \frac{n}{2}$. In this case,

$$I_m = f_{m+1} + f_{n-m+2} > f_{n-m+2} \geq I_{\frac{n}{2}}, \text{ where the inequality is strict when } m < \frac{n}{2} - 1.$$

Similarly, it can be shown that the search space is greater than $I_{\frac{n}{2}}$ for the case of $m > \frac{n}{2}$. Note that, while only the reduction in the search space for $k = 1$ is illustrated, the same can also be verified for $k \geq 2$. For further details, refer to Krafft and Schaefer (2002).

B.2 Reduction in B&B nodes exploration

Table B.1 B&B nodes comparison between DS-ASP and ASP

	Node Count						% Reduction
$k \rightarrow$	1		2		3		
F	DS-ASP Avg. (Range)	ASP Avg. (Range)	DS-ASP Avg. (Range)	ASP Avg. (Range)	DS-ASP Avg. (Range)	ASP Avg. (Range)	
30^d	651 (362-1,434)	7,198 (4,770-11,830)	25,523 (14,263-47,390)	208,044 (30,891-574,881)	120,604 (58,143-210,978)	>1,949,693 (366,831->4,825,865)	>93.2
40^d	3,190 (152-11,258)	10,834 (1,792-54,693)	102,229 (11,268-204,461)	>1,659,600 (50,893->5,285,228)	450,329 (35,158-1,021,030)	>4,188,216 (1,291,210->8,773,352)	>90.5
30^m	632 (122-1,270)	5,120 (1,744-10,171)	48,960 (14,938-85,062)	312,792 (57,699-688,804)	296,416 (96,757-522,622)	>2,899,895 (192,632->8,254,418)	>89.2
40^m	2,937 (1,096-3,507)	7,824 (1,668-17,304)	214,109 (146,540-238,984)	>4,102,956 (1,857,021->6,281,425)	4,498,131 (1,200,400-12,855,984)	>4,999,556 (>4,516,818->6,405,605)	>48.3

Appendix C

Pseudo-Code for multiple-split

Initialize: $S \leftarrow \mathcal{F}$; $\hat{z}_1, \hat{z}_2 \leftarrow 0$; $\hat{l}_1, \hat{l}_2 \leftarrow \text{dummy}$; $\eta_1, \eta_2, \eta_3 \leftarrow 0$. Call Data-Splitting ($S, \hat{z}_2, \hat{z}_1, \hat{l}_2, \hat{l}_1, \eta_1, \eta_2, \eta_3$) to get the best solution. Note that $\eta_3 = 0$ indicates call after the initial leading set get processed, $\eta_3 = 1$ (resp. $\eta_3 = 2$) indicates call after intermediate set get processed with makespan higher on first runway (resp. second runway).

function DATA-SPLITTING ($S, \hat{z}_2, \hat{z}_1, \hat{l}_2, \hat{l}_1, \eta_1, \eta_2, \eta_3$)

- 1: Set iteration counter $p \leftarrow 0$, $OptVal^{(p)} \leftarrow +\infty$, incumbent $\leftarrow OptVal^{(p)}$, split S to give $\mathcal{P} = \{\text{set of all pairs } (\ddot{A}, B)\}$ with $|\ddot{A}| = n_1 \leq \phi_1$, and $|B| = n_2$.
- 2: If $\mathcal{P} = \emptyset$, stop; return incumbent (and incumbent δ -value in case $\eta_1 = 1$ or $\eta_2 = 1$) as the best solution; else, set $p \leftarrow p + 1$; go to Step 3.
- 3: Arbitrarily select one instance pair $(\ddot{A}_p, B_p)$ from $\mathcal{P}$. Set temp.best, $\ddot{z}_2 \leftarrow +\infty$.
- 4: **if** $\eta_3 = 0$ **then**
- 5: Solve $Opt_{\ddot{A}_p}$. Go to Step 11.
- 6: **else if** $\eta_3 = 1$ **then**
- 7: Solve $Opt_{\ddot{A}_p}^1$. Go to Step 11.
- 8: **else if** $\eta_3 = 2$ **then**
- 9: Solve $Opt_{\ddot{A}_p}^2$. Go to Step 11.
- 10: **end if**
- 11: Let $(\ddot{z}_1, \ddot{z}_2)$ be a set of optimal solution (and δ_p is δ -value, if $\eta_1 = 1$ or $\eta_2 = 1$).
- 12: **if** $n_2 > \phi_2$ **then**

- 13: **if** $\eta_3 = 0$ or $\eta_3 = 1$ **then**
- 14: Set $\eta_1 \leftarrow 1$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 1$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $(\delta, \bar{z}_\delta)$. If $\delta > 0$ go to Step 15; else set $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 15: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 1$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $\bar{z}$. If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 16: Set $\eta_1 \leftarrow 1$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 2$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $(\delta, \bar{z}_\delta)$. If $\delta > 0$ go to Step 17; else set $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 17: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 2$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $\bar{z}$. If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 18: **else if** $\eta_3 = 2$ **then**
- 19: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 1$, and $\eta_3 \leftarrow 1$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $(\delta, \bar{z}_\delta)$. If $\delta > 0$ go to Step 20; else set $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 20: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 1$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $\bar{z}$. If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 21: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 1$, and $\eta_3 \leftarrow 2$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $(\delta, \bar{z}_\delta)$. If $\delta > 0$ go to Step 22; else set $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 22: Set $\eta_1 \leftarrow 0$, $\eta_2 \leftarrow 0$, and $\eta_3 \leftarrow 2$. Call *Data-Splitting* ($B_p, \check{z}_1, \check{z}_2, l_1, l_2, \eta_1, \eta_2, \eta_3$) to get $\bar{z}$. If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 23: **end if**
- 24: **else**
- 25: **if** $\eta_3 = 0$ **then**
- 26: Solve $\delta - \text{Opt}_{B_p}$. If $\delta > 0$ go to Step 31; else $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 27: Solve Opt_{B_p} . If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 28: **else if** $\eta_3 = 1$ **then**
- 29: Solve $\delta - \text{Opt}_{B_p}^1$. If $\delta > 0$ go to Step 34; else $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 30: Solve Opt_{B_p} . If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 31: **else if** $\eta_3 = 2$ **then**
- 32: Solve $\delta - \text{Opt}_{B_p}^2$. If $\delta > 0$ go to Step 37; else $\bar{z} \leftarrow \bar{z}_\delta$, go to Step 36.
- 33: Solve Opt_{B_p} . If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$. Go to Step 4.
- 34: **end if**
- 35: **end if**
- 36: If $\bar{z} < \text{temp.best}$, set $\text{temp.best} \leftarrow \bar{z}$; $\text{OptVal}^{(p)} \leftarrow \text{temp.best}$.
- 37: If $\text{OptVal}^{(p)} < \text{incumbent}$, set incumbent as $\text{OptVal}^{(p)}$ and $(\delta_p$ as incumbent δ -value, if $\eta_1 = 1$ or $\eta_2 = 1$) . Update $S \leftarrow S \setminus \{\check{A}_p, B_p\}$. Go to Step 2.

Appendix D

Publications Resulting from this Thesis

1. Prakash, R., Piplani, R., and Desai, J. (2018). An optimal data-splitting algorithm for aircraft scheduling on a single runway to maximize throughput. *Transportation Research Part C: Emerging Technologies*, 95:570 – 581.
2. Prakash, R., Desai, J., and Piplani, R. (2018). An optimal data-splitting algorithm for aircraft sequencing on a single runway. *Working Paper*, Nanyang Technological University Singapore.
3. Prakash, R., Piplani, R., and Desai, J. a. (2018). An optimal data-splitting algorithm for aircraft sequencing on two runways. *Working Paper*, Nanyang Technological University Singapore.
4. Prakash, R. and Desai, J. (2017), “A data-splitting algorithm for flight sequencing and scheduling on two runways”, *Proceedings of the Industrial & Systems Engineering Research Conference*, Pittsburgh, USA.
5. Desai, J. and Prakash, R. (2016), “Flight sequencing and scheduling: A data-driven approach”, *Proceedings of the Industrial & Systems Engineering Research Conference*, Anaheim, USA.
6. Desai, J. and Prakash, R. (2016), “An optimization framework for terminal sequencing and scheduling: The single runway case”, *Proceedings of the Complex Systems Design & Management Conference, In: Advances in Intelligent Systems and Computing (Springer)*, Singapore, 195-207.